Object Detection and Defense against Adversarial Images

Ashfaque Azad

MSc in Computer Science (Data Analytics)

School of Engineering & Informatics

National University of Ireland, Galway

Supervisor:

Prof. Michael G. Madden

AUGUST 2018

Contents

Acknowledgments

I would like to thank my supervisor Prof. Michael G. Madden for his guidance, patience, and encouragement throughout my thesis work.

I would like to thank my parents for their support all through my postgraduate degree.

Abstract

This thesis first investigates the challenge of identifying the presence of an item (e.g. a human or a road-sign) within a complex image i.e. an image containing multiple objects; second, it probes a simple strategy that defends against adversarial inputs, which are carefully calculated perturbation introduced on images purposefully to misclassify or “fool” state-of-the-art image classifiers, on Deep Neural Networks (DNNs) for Image Recognition. DNNs are discussed in the later chapters.

For the purpose of object detection we study in details the working of YOLO (You Only Look Once), one of the current object detection approaches, which is based on deep neural networks. Next, we look into adversarial examples, its meaning, generation, impact and develop a simple strategy using input modifications and ensemble network of pre-trained image classifiers for defending against adversarial examples.

A new attack system is also proposed whereby a single adversarial image can fool two different pre-trained models simultaneously.

1. Introduction

Digital images, in the context of deep learning, are just an array of numbers. These images sometimes come with one channel, sometimes with three different channels (e.g. RGB : red, green, and blue color-channels) where each pixel is represented by three different numbers [1].

The goal of computer vision is to replicate the function of the human visual system [2]. Computer vision is widely used in industries such as autonomous driving.

One of the major tasks in the field of deep learning and computer vision is image classification. A few of the challenges that faces image classification are illumination variability (the recognition of objects under different lighting conditions pose a challenge for object detection), pose variability (an object's pose in the plane of color and texture are quite different numerically when it is posed differently at different angles), intra-class variability (there may be different types of the same object, for example, different breeds of cat or dog), and occlusion (partial visibility or hidden objects i.e. objects partially hidden behind other objects) [1].

Convolutional Neural Networks (CNNs), discussed in details in Chapter 2, due to the property of spatial invariance (the idea that an object, say in the top-left corner of an image, is the same as at any other space in the image) may throw away the hierarchical information relationship between simple and complex objects, e.g. an image of a face with an eye outside the face, or nose at the position where lips should be, will still be classified as a face [1].

1.1 Problem statement

Object detection is the process of finding real-world objects such as vehicles, person, traffic signs in images or videos while image classification is used to classify an image into a particular category; object detection is the localization of these categories within an image [1].

Deep Neural Networks are shown to be vulnerable to adversarial perturbations, which when overlaid on an image, can cause a classifier to misclassify with high confidence. These perturbations should be such that they are imperceptible to humans [3].

Attacks can be one of the three types: black, white, and gray [4]. In the black-box attack scenario, the adversary does not have access to the network parameters; in the grey-box attack, the adversary does not have access to the defense mechanism but has access to the network weights; in the case of the white-box attacks, the adversary knows about the weights of the network as well as about the defense mechanism [4].

All of the pre-trained models which have been worked on with in this thesis have been trained on the ImageNet [5] dataset. ImageNet is a dataset of more than 14 million images with over 21 thousand categories of objects [5]. YOLO (details in chapter 3), on the other hand has been trained on the COCO dataset (see appendix).

Various defenses have been attempted against these crafted perturbations. Two of the significant defenses are adversarial training [6] and defense distillation [7]. Adversarial training is a solution where a lot of adversarial examples are generated and a model is trained explicitly with them to not get fooled by adversarial inputs [6]. It is a brute-force solution. In case of Defense Distillation, a model is trained to output the class probabilities instead of hard decisions about which class to output; these probabilities are supplied by an earlier model which is trained on the same task using hard-class labels [8].

Even though these are quite good defenses, yet they have been bypassed by crafting more robust adversarial images [9][10].

1.2 Motivation

YOLO (You Only Look Once) is one of the real-time object detection system. It uses a single neural network on an entire image; it then divides the image into regions and predicts bounding boxes and probabilities for each region [11]. YOLO is extremely fast object-detection system [11]. This thesis focuses on how it works, and points out its shortcomings.

Re-training a network is quite expensive and time-consuming. It is almost like building stronger walls around vulnerable castles, but there may be attacks which are stronger and which may result again in having to re-train the network. As it is already seen that many of these defenses have already been penetrated by stronger attacks [10].

This thesis focuses on how input transformations before inputting the image into an ensemble of classifiers can defend against adversarial examples, it also introduces an attack method which fools two pre-trained DNNs at the same time.

This relies upon- how simple transformations such as translations or rotations are sufficient enough to fool image-classifiers [12]. It also uses a simple local smoothing technique to smooth pixels [13]. It is done so as to blur out the effect of adversarial inputs on the images. These input transformations are over-simplistic in nature, and have already been bypassed by using robust adversarial attacks [14]. Our proposed strategy was to use a) multiple of these simplistic defenses like local smoothing and rotations and b) then feeding the modified input into an ensemble of different pre-trained image classifiers. This ensemble is based on the idea of prediction inconsistency which states that one adversarial example may not fool every DNN model [13].

A new attack system is also proposed in this thesis, where the ensemble (without the input modifications as the first line of defense) of pre-trained models had not only one but two of its pre-trained models fooled by a single attack.

1.3 Research Questions

- How do the current state-of-the-art object detection systems perform in terms of speed and accuracy?
- What are the shortcomings of YOLO object detection system?
- Can there be a simple solution for defending against adversarial examples which do not require retraining a neural network?

2. Literature Review

2.1 Neural Network

The goal of a neural network is to approximate some function f^*[15].Let x be the input and y be the output. A feedforward neural network defines a mapping $y=f(x;\theta)$ and learns the parameters θ that results in the best function approximation[15].

These feedforward neural networks form the basis of many important applications, for example, object detection from photos require special kind of feed forward neural networks called convolutional neural network [15].

Let us say that we have three functions $f^{(1)}$, $f^{(2)}$, $f^{(3)}$ connected in a chain to form $f(x)= f^{(3)}(f^{(2)}(f^{(1)}(x)))$; $f^{(1)}$, $f^{(2)}$, $f^{(3)}$ are the first, second, and third layers of the network respectively [15]. The total length of the chain gives the depth of the model, with the final layer being the output layer, a neural network is trained to match $f(x)$ to $f^*(x)$ [15].

Deep learning provides a powerful framework for supervised learning. By adding more layers or units within those layers of a neural network, a deep network may represent functions of increasing complexity [16].

2.2 Deep Learning aspect for Computer Vision

Traditional methods for computer vision and machine learning cannot match human performance on recognition tasks of handwritten words or traffic signs, however deep neural architectures can achieve an accuracy close to the human performance [17].

The British Machine Vision Association and Society for Pattern Recognition defines computer vision as a field "*concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding* [18]."

Computer vision has its use in numerous applications some of which are listed below:

1. *Autonomous vehicles* : object detection is a crucial component for autonomous vehicles [18].
2. *Face recognition* : identifying a person's face from a digital image or a video feed [18].
3. *Surveillance* : computer vision is indispensable in surveillance activities, for example, vehicle tracking [18].

2.2.1 Convolutional Neural Networks (CNNs):

CNNs are a specialized kind of deep neural network for processing data that has a known grid-like topology, one of the examples being image data [19].

A CNN consists of three layers typically[19] :

1. Convolutional
2. Pooling
3. Fully connected

"Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. Convolution is an operation on two functions of a real-valued argument [19]."

Figure 2.1 shows the convolution operation. *The boxes with arrows indicate how the upper-left element of the output tensor is formed by applying the kernel to the corresponding upper-left region of the input tensor* [19].

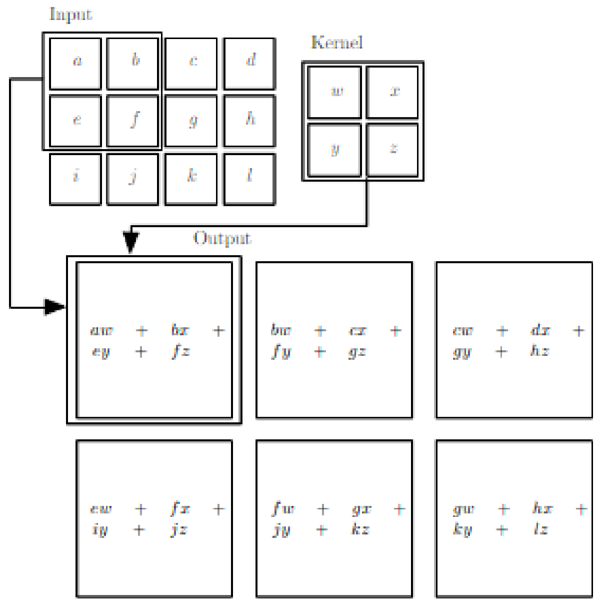

Fig.2.1: 2-D convolution (source: **deeplearningbook.org** [19])[1]

The pooling function replaces the output of a net at a certain location with a summary statistic of the nearby outputs, for example with max pooling the operation reports the maximum output within a rectangular neighborhood [19].

The last fully connected layer contains neurons which equals in the number of the total classes to be predicted [19].

Object recognition for autonomous driving:

Autonomous vehicles have the potential to significantly reduce the number of fatalities due to car accidents. The recent self-driving car project at Google [20] has shown that detailed and highly-accurate object recognition is largely not required. However the fatal accident in Arizona [21] has put a dent in the autonomous driving industry, leading to questions like: is mere object detection sufficient to get autonomous cars on the roads? Google's autonomous driving

[1] Goodfellow I.,Bengio Y.,Courville A.(2016), "Chapter 9 : Convolutional Networks" https://www.deeplearningbook.org/contents/convnets.html

systems are based on building detailed maps of the world, then localizing them during operation [20]. As a result, construction zones present a challenge here. Object detection models used in autonomous driving should not just detect a human but also recognize a person's cab hailing sign gesture for autonomous taxis, for example [22].

2.3 Adversarial Examples

Neural networks may become unstable when small perturbations are introduced to their inputs; adversarial examples are carefully crafted perturbations designed to make deep neural networks misclassify [23].

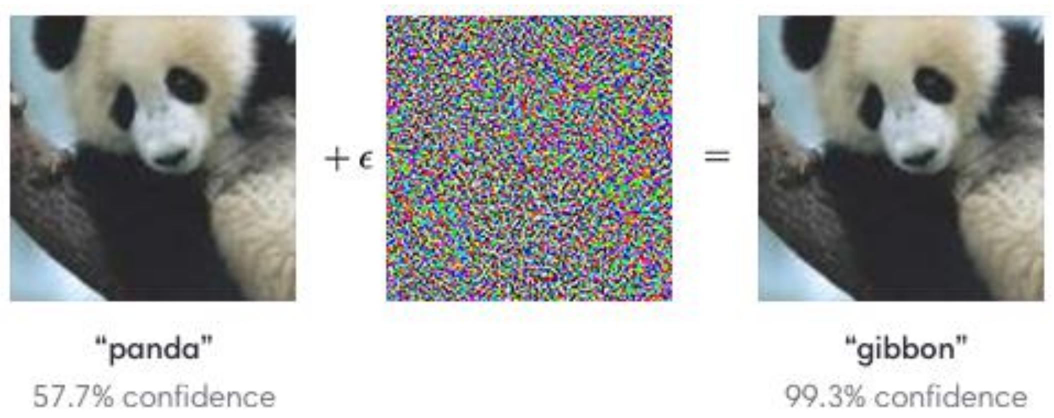

Fig. 2.2 : demonstration of adversarial example(source: **Explaining And Harnessing Adversarial Examples** [3])[2]

Figure 2.2 shows that by adding small perturbation to the image of a panda the image may be classified by the attacked neural network as a gibbon with high confidence [23]. ϵ corresponds to the smallest bit's magnitude of an 8-bit image encoding [3].

Adversarial images have the potential to be dangerous, as it may cause the image classifiers to get fooled and help in fraud evasion, for example.

[2] Goodfellow I.,Shlens J.,Szegedy (2014) ," Explaining and Harnessing Adversarial examples". *International Conference on Learning Representations 2015.* https://arxiv.org/pdf/1412.6572.pdf

i. Types of adversarial attacks:

Let R be the robustness of a model, which is defined as the average size of the minimum adversarial perturbation $\rho(x)$ across many samples x,

$R = \langle \rho(x) \rangle_x$ where (1 source [24])

$\rho(x) = \min_{\delta} d(x, x+\delta)$ s.t. $x+\delta$ is adversarial (2 source [24])

and d(.) is some distance measure [24].

Notations [24]**:**

x	a model input
l	a class label
x_0	reference input
l_0	reference label
$L(x,l)$	loss (e.g. cross-entropy)
$[b_{min}, b_{max}]$	input bounds

1. **Gradient-based attacks:**

These type of attacks linearize the loss around an input x to find directions ρ to which the model predictions for class l are most sensitive to,

$L(x+\rho,l) \equiv L(x,l) + \rho^T \nabla_x L(x,l)$ (3 source [24])

$\nabla_x L(x,l)$ is the gradient of the loss w.r.t the input x [24].

Example : L-BFGS-B attack (explained in the chapter 4).

2. **Score-based attacks:**

These type of attacks do not require gradients of the model, but they expect probabilities or logits which can be used to approximate gradients [24].

Example : Single-pixel attack. *This attack probes the robustness of a model to changes of single pixels by setting a single pixel to black or white; it repeats this process for every pixel in the image* [24].

3. **Decision-based attacks:**

These type of attacks rely on the class decision of the model; they require neither probabilities nor gradients [24].

Example : Salt and pepper noise attack. *This minimizes the L2-norm of adversarial perturbations [24].*

3. Object Detection

While an image classifier classifies an image into a certain category, object detection helps identifying the location of an object within an image [25].

In order to train a model, we'd require labeled data. In context of object detection, these data are images with corresponding bounding box coordinates and labels , this is illustrated in figure 3.1.

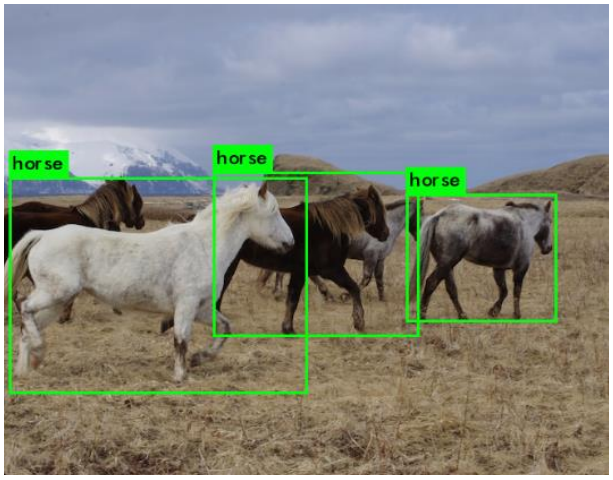

Fig 3.1: object detector surrounds the object with bounding boxes and labels them (source : **YOLO** [11])[3]

3.1 YOLO

You Only Look Once (YOLO) is a real-time object detection system; object detection algorithms, prior to YOLO, applied the models to an image at various locations and scales, the regions which scored high were considered detections [11].

Before we see how YOLO works, we discuss briefly some of the other detection systems.

[3] YOLO. Retrieved from https://pjreddie.com/darknet/yolo/ on 21st July 2018.

Other object detection systems:

Deformable Parts Model: uses sliding window approach; here the classifier is run at evenly spaced locations over the entire image [26]. It uses a disjoint pipeline to extract static features, classify regions, predict bounding boxes for high scoring regions et cetera [26].

R-CNN: uses region-based proposal methods to generate bounding boxes in an image and then run a classifier on these boxes; selective search generates bounding boxes, a CNN extracts features, an SVM (Linear Support vector machine is a supervised learning model which finds maximum margin function that linearly separates two classes; it follows the intuition of placing a hyperplane as “far” as possible from known data of both classes [27]) is used to score the boxes, a linear model adjusts the bounding boxes, and a non-max suppression removes duplicate detections [26]. This is a slow process as each stage of this pipeline must be precisely tuned independently [26].

YOLO, on the other hand, applies a single neural network to the entire image [11]. The image is divided into regions called grids or cells, which predicts bounding boxes and probabilities for each region [11]. These bounding boxes are weighted by predicted probabilities [11].

‘YOLOv2 608x608’ is the model, trained on COCO dataset (see appendix), used in this thesis.

The model works in the following way [26]:

1. Resizes the input image to 608x608
2. Runs a single convolutional network on the image
3. Thresholds the resulting detections by the model’s confidence.

Fast R-CNN (Fast Region-based convolutional network method is another object detection system which trains very deep VGG16 9-times faster than R-CNN, and is 213-times faster at test time [28]) fails to see the larger context and mistakes background patches within an image for objects [26]. YOLO makes less than half the errors in the same situation; YOLO being highly generalizable, is less likely to break down when applied to new domains, such as art-work (an example of which is presented in the figure 3.2 where the painting of Mona Lisa (the subject- Mona Lisa being the object) is correctly classified as a person) [26].

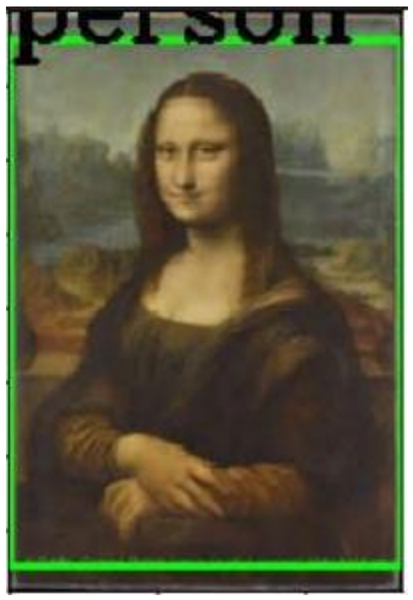

Fig. 3.2. YOLO detects the Mona Lisa to be a person, this raises the question that it may be unable to differentiate between a real human being and a painting of a person. (source : **Mona Lisa, Louvre** [29])[4]

However YOLO lags behind state-of-the-art object detection systems in terms of accuracy, and localizing objects especially smaller ones within an image [26].

How it works:

YOLO divides the input image into S x S grid [26]. If the centre of an object falls in one of the grid cells, that cell is responsible for detecting the object [26]. Each cell predicts B bounding boxes and confidence scores for those boxes, if no object exists within a cell, the confidence should be zero [26]. This is illustrated in figure 3.3.

[4] Mona lisa ,Louvre. Retrieved from https://www.louvre.fr/en/mediaimages/portrait-de-lisa-gherardini-epouse-de-francesco-del-giocondo-dite-monna-lisa-la-gioconda on 11th August 2018.

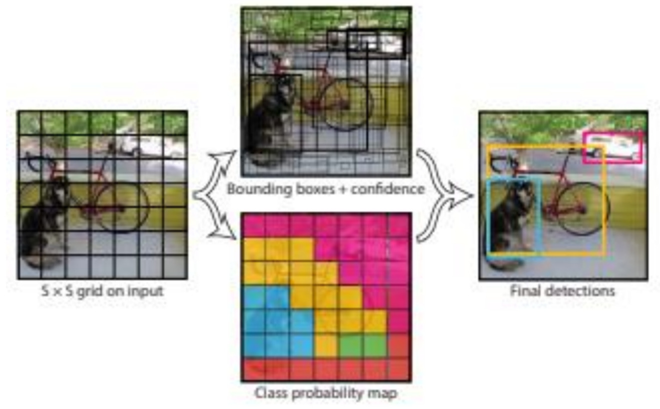

Fig 3.3 :The image is divided into SxS grid, for each of the grids B number of bounding boxes are predicted, along with confidences for those boxes and C class probabilities.(source : **You Only Look Once: Unified, Real-time Object Detection** [26])[5]

Each bounding box gives 5 predictions - x,y,w,h, and confidence [26].

(x,y) : represents the center of the box relative to the bounds of the grid cell;

(w,h) : width and height predicted relative to the whole image;

confidence : intersection over union (IOU) between the predicted box and the ground truth [26].

Each cell also predicts C (conditional class probabilities) : $Pr(Class_i|Object)$. At test time conditional class probabilities is multiplied with the individual box confidence predictions [26]. This score gives us the probability of the class appearing in the box and how well the box fits the object; the initial convolutional layers extract features from the image while the fully connected layers predict the output probabilities and coordinates [26].

For this thesis darkflow [30]: tensorflow version of darknet (open source neural network framework written in C and CUDA (*parallel computing platform and application programming interface model* [31])) implementation of YOLOv2 is used. The pre-trained weights are downloaded from [32].

[5] Redmon J.,Divvala S.,Girshick R.,Farhadi A.(2015),” You Only Look Once: Unified, Real-Time Object Detection” https://arxiv.org/pdf/1506.02640.pdf

The threshold probability is set at 0.4. This is the bottom line of the confidence values chosen for keeping the detected objects. Any value below that is rejected, as any value less than 40% is taken as not too confident prediction by YOLO. For the purpose of this project every object that it (YOLO) detects, a bounding box of colour green is used to surround that object and label it with the class of the object detected. This is observed in the figure 3.2.

A few examples illustrated using YOLOv2 608x608 as a part of this thesis:

The purpose of these experiments was to illustrate the working of YOLO [26] on a number of images taken of roads containing objects like traffic signals, road signs, people, vehicles et cetera.

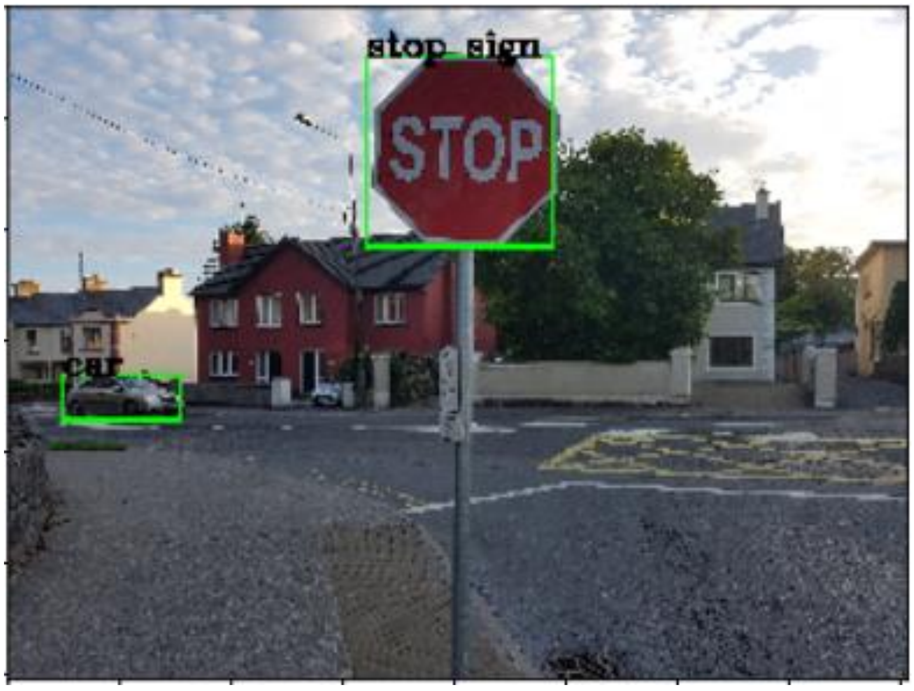

Fig. 3.4: YOLO detected 2 objects in this image

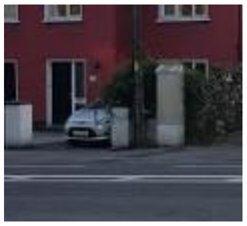

Fig. 3.5: the car in front of the house which goes undetected by YOLO

The tables for the corresponding images represent if the object is detected and gives its corresponding confidence.

Table for figure 3.4.

Objects Present	**Detected?**	**Confidence%**
STOP SIGN	YES	89.2
CAR	YES	81.9
CAR	NO	-

Figure 3.4 has a small car visible in between the two detected objects in front of the red-colored house (figure 3.5), which goes undetected by YOLO. This illustrates the point mentioned in [26] which states YOLO struggles to localize some objects, especially small ones.

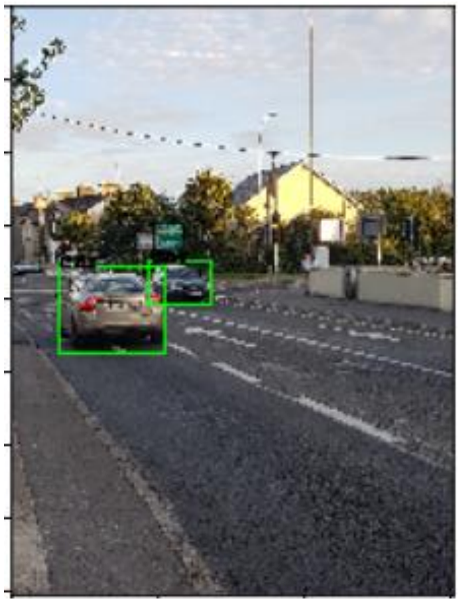

Fig. 3.6.

Table for figure 3.6.

Objects Present	**Detected?**	**Confidence%**
VISIBLE TRAFFIC SIGNALS	NO	-
CAR	YES	79.5
CAR	YES	71.6
CAR	NO	-
CAR	NO	-

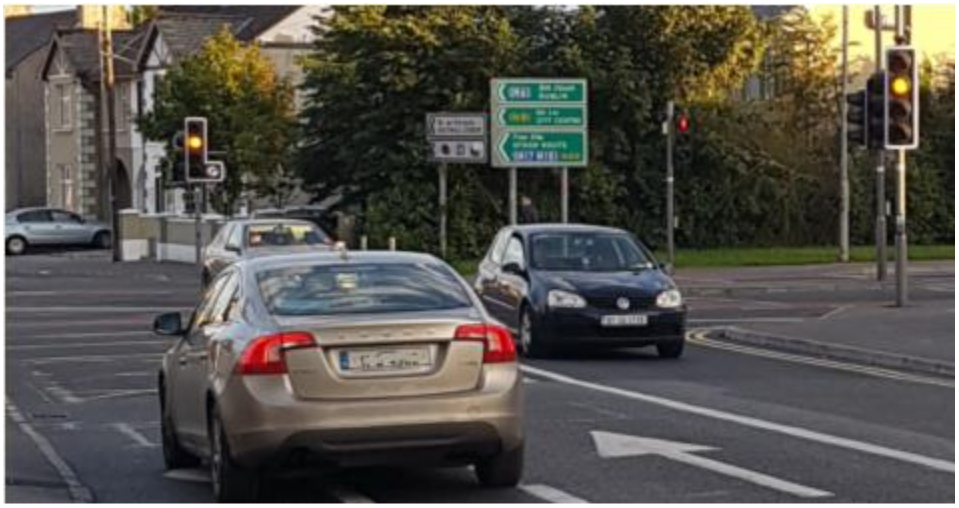

Fig. 3.7: more focused (zoomed-in) version of the selected part of the figure 3.6

YOLO could detect only 2 of the 4 vehicles present in the image (figure 3.7). It also fails to detect the far off traffic signals.

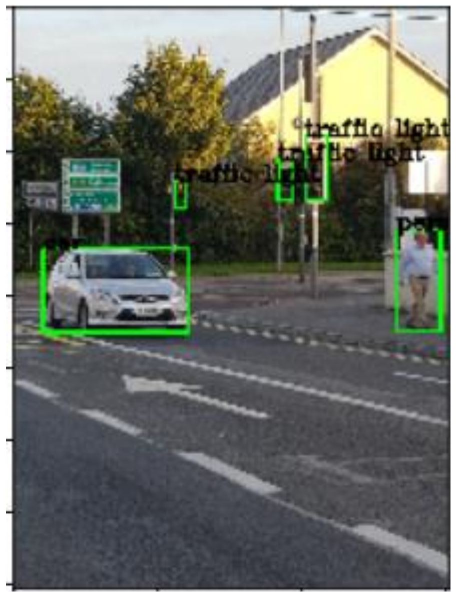

Fig. 3.8: this image was clicked at same location as in figure 3.6 but closer by few meters to the traffic signal.

Table for figure 3.8.

Objects Present	**Detected?**	**Confidence%**
VISIBLE TRAFFIC SIGNAL	YES	66.4
VISIBLE TRAFFIC SIGNAL	YES	50.7
CAR	YES	77.3
PERSON	YES	71.7
HIDDEN TRAFFIC LIGHT	YES	44.2

Upon getting an image closer to the traffic signals by few meters (compared to the one in Figure 3.6), we are able to see that YOLO can detect not only all traffic signals but also the ones facing not directly at the camera.

It has been seen that YOLO can read the STOP sign quite accurately. Next was to test out if it could read other signs as well.

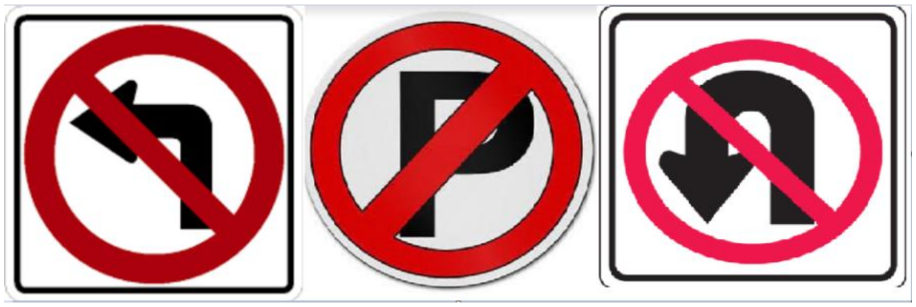

Fig. 3.9:road signs(source : **epermittest.com/road-signs** [33])[6]

Figure 3.9. shows three road signs : no left turn, no parking, and no u-turn.

YOLO fails to recognize these as street signs, moreover it fails to recognize them as any sign at all. However it perfectly predicts the stop sign as shown in figure 3.10.

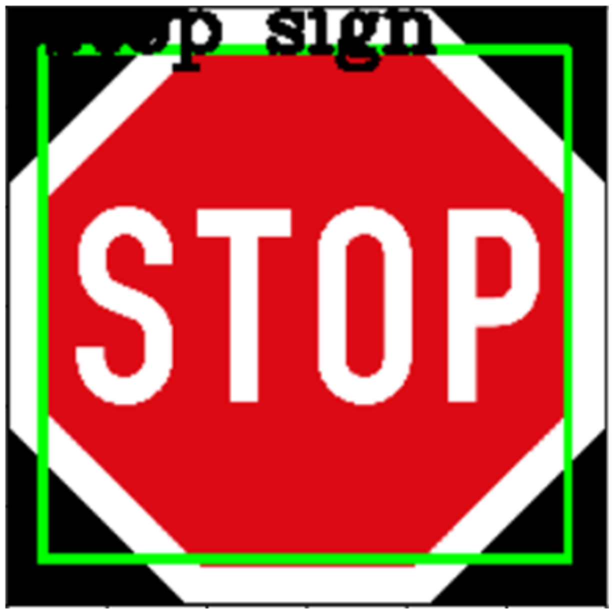

Fig. 3.10: stop sign(source : **epermittest.com/road-signs** [33])[7]

YOLO predicts the image in figure 3.10 as a STOP sign with 64% confidence.

[6] Road signs . Retrieved from http://www.epermittest.com/road-signs on 1st August 2018.
[7] Road signs . Retrieved from http://www.epermittest.com/road-signs on 2nd August 2018.

4. Targeted Adversarial Attack

The adversarial image generation algorithm mentioned in this chapter is published in the paper “Exploring the Space of Adversarial Images [34].”

4.1 Creating Adversarial Images :

Let p=$f(X)$ be a pre-trained classifier, where X is the input [34].

L , U correspond to the lower and upper limit of the pixel scale [34].

I = [L-U] for grayscale images [34].

I= $[\text{L-U}]^3$ for RGB images [34].

$X \in I$, I is the pixels of the fixed size image [34].

The classifier outputs the vector of probabilities p = $[p_1 \ldots p_i \ldots p_n]$ of the image belonging to the class i [34].

h is assigned to the label corresponding to the highest probability p_h [34].

Assume c as the correct label, and h=c.

The aim is to introduce a distortion D to X, such that the highest probability will no longer be assigned to h [34].

It must be kept in mind that $X + D \in I$ [34]. This means that the input is box constrained.

L-BFGS-B (the algorithm being discussed here) minimizes the distance between the image and the adversarial; it also minimizes the cross-entropy between the predictions for the adversarial and the one-hot encoded target class [24].

We have the following optimization :

$$\begin{aligned} &\text{minimize}_D && ||\, D \,|| \\ &\text{subject to} && L \le X + D \le U && \text{(1 source [34])} \\ &&& p = f(X + D) \\ &&& \max(p_1 - p_c \,.....\, p_n - p_c) > 0 \end{aligned}$$

4.2.1 Procedure:

In generating adversarial images the model weights are held fixed [34].

Let H be the cross-entropy.

p^A be the adversarial probability target which assigns zero probability to all but the chosen adversarial label a [34].

$p^A = [1_{i=a}]$ [34].

$$\begin{aligned} &\text{minimize}_D && ||\, D \,|| + C.H(p, p^A) \\ &\text{subject to} && L \le X + D \le U && \text{(2 source [34])} \\ &&& p = f(X + D) \end{aligned}$$

C must be lowest and yet feasible [34].

The optimization (equation 2) variables are the pixel distortions; hence the problem size is exactly the size of the network input. For generating adversarial, pre-trained keras (see appendix) model ResNet50 was used which has 150528 (224*224*3) as its input.

The model being attacked to generate adversarial examples using this algorithm is ResNet50 : a 50-layer convolutional neural network based on residual learning, where the layers higher up in the network architecture skips some connections and connects to the downstream layers to improve performance [35].

L-BFGS-B (the algorithm in section 4.2.2)is a limited memory algorithm for solving nonlinear optimization problems subject to bounds on the variables [36].

4.2.2 Algorithm :

A bisection search is performed to search for the constant C [34].

An initial upper and lower bound for C is required for bisection, in such a way that the upper bound successfully finds an adversarial image and the lower bound does not [34].

It then searches for a transition point from failure to success that will be the best C [34].

[This algorithm (below) is the work of **Pedro Tabacof** and **Eduardo Valle** [34].*]*

```
requirement: a small positive $\epsilon$
ensure: L-BFGS-B($X,p^A,C$) solves eq$^n$ 2.
$C$ <- $\epsilon$
repeat
        $C$ <- $2*C$
        $D,p$ <- L-BFGS-B($X,p^A,C$)
until max($p_i$) in $p$ is $p_a$
$C_{low}$ <- 0 , $C_{high}$ <- 0
repeat
        $C_{half}$ <- $(C_{low} + C_{high})/2$
        $D', p$ <- L-BFGS-B($X,p^A,C_{half}$)
        if max($p_i$) in $p$ is $p_a$ then
```

```
            $D$ <- $D'$
            $C_{high}$ <- $C_{half}$
        else
            $C_{low}$ <- $C_{half}$
        end if
until ($C_{high} - C_{low}$) < $\epsilon$
return $D$
```

[algorithm ends]

Experiments:

The purpose of the following experiments is to use the algorithm as mentioned in [34] and generate adversarial examples. The adversarial generating algorithm attacks the RESNET50 model and classifies the adversarial image in figure 4.1 as that of an airliner.

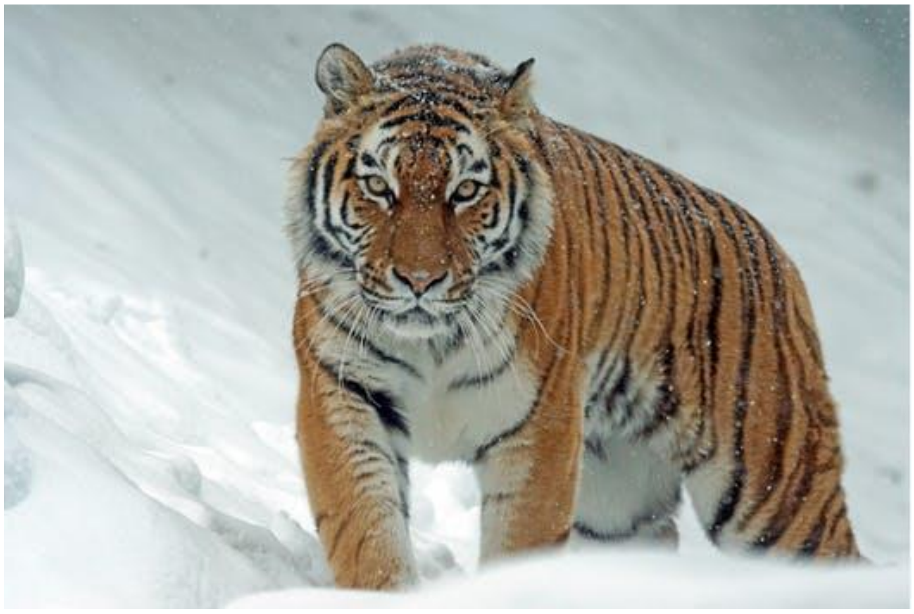

Fig. 4.1: original unperturbed image of a tiger (source: **pexels.com/search/tiger** [37])[8]

[8] Image. Retrieved from https://www.pexels.com/search/tiger/ on 21st July 2018.

The target adversarial class chosen was an airliner.

The target criteria was set at 85%. This means that the perturbations being generated on the original image of a tiger by the algorithm would cease when the model (being attacked) classifies it as an airliner with confidence 85% or more.

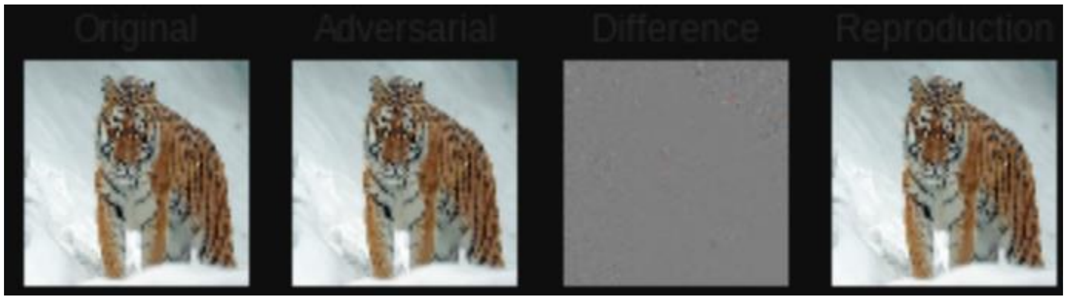

Fig. 4.2 (From left to right: Original, Adversarial, Difference, and Reproduction.)

Original : Unperturbed image of a tiger.

Adversarial: Perturbed image classified as airliner by ResNet50.

Difference: Distortions (magnified in the figure) is the difference between original and adversarial images.

Reproduction: is the regeneration of the original image by subtracting the distortions or perturbations from the adversarial image.

The purpose of the image labeled 'Reproduction' is just to see if the model has the same confidence as that of the 'Original' image.

The table below shows the model's prediction on the images in Figure 4.2.

IMAGE	**PREDICTED CLASS**	**CONFIDENCE %**
ORIGINAL	TIGER	68.9
ADVERSARIAL	AIRLINER	92.5
REPRODUCTION	TIGER	68.9

The reproduced image has the same confidence as that of the original image.

Fig. 4.3: image of a daisy.

Figure 4.3. is the unperturbed image of a daisy.

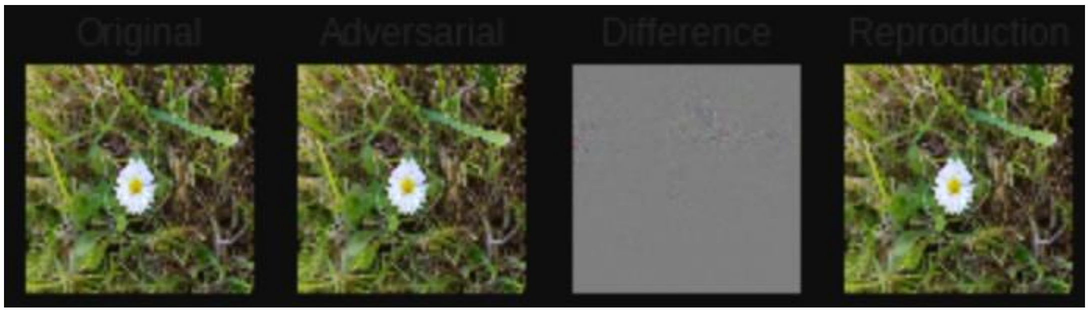

Fig. 4.4: From left to right : Original, Adversarial, Difference, and Reproduction.

Here too the targeted adversarial class was set at airliner with confidence set at 85% as before.

The table below shows the model's prediction on the images in Figure 4.4.

IMAGE	**PREDICTED CLASS**	**CONFIDENCE %**
ORIGINAL	DAISY	46.7
ADVERSARIAL	AIRLINER	89.6
REPRODUCTION	DAISY	46.7

In both the cases we see the adversarial or the perturbed images easily fooling the network with much higher confidence with respect to the confidence related to that of the original images.

5. Brief discussion about the models used in the ensemble

The goal of ensembles is to use multiple classifiers (in our case, image classifiers) and combine their predictions/decisions to arrive at a final conclusion [27].*As long as individual classifiers are independent and better than random, **ensemble** as a whole will be better than any one of the individuals, on average, and never significantly worse* [27].

The defense technique used in this thesis uses exactly this concept of an ensemble method. VGG16, RESNET18, and DENSENET are the pre-trained (on ImageNet) image-classifier models used together as an ensemble here.

Sections 5.1, 5.2, and 5.3 briefly explains the structure of these classifiers.

5.1. VGG16

Architecture:

Input to the convolutional neural network is a fixed-sized 224x224 RGB image, the preprocessing is done in this way : the mean RGB value, computed on the training set, is subtracted from each pixel [38]. It takes values in the range of 0-255; the image is then passed through convolutional layers where 3x3 filters are used. Max-pooling is performed over a 2x2 pixel window of stride 2 [38].The stack of convolutional layers is followed by three fully-connected layers, which is then followed by a final softmax layer; all hidden layers are equipped with activation function- ReLU [38].

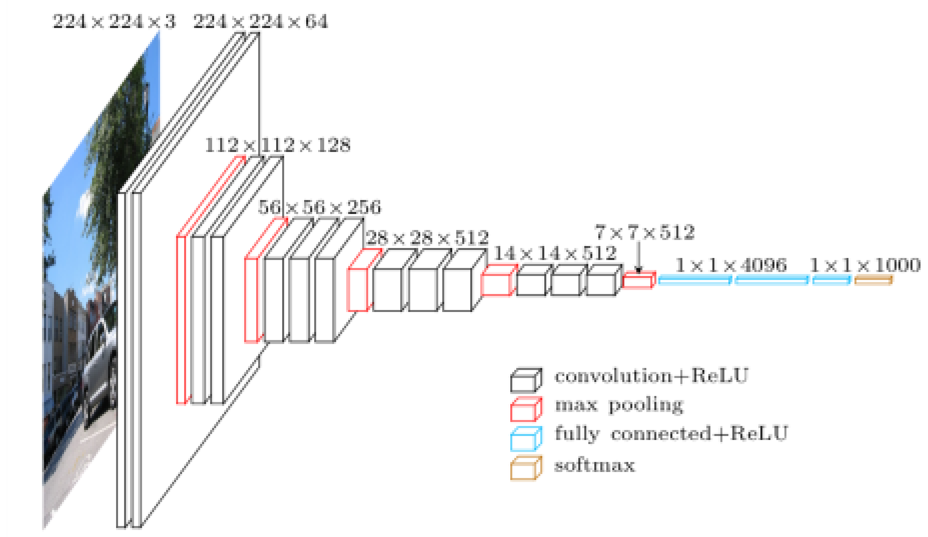

Fig. 5.1: macro-architecture of VGG16 (source : **Heuritech** [39])[9]

Even though this architecture has long depth of nets, the number of epochs required for convergence is less [38]. VGG16 for this thesis, is trained on the subset of the ImageNet database; it can classify images of one-thousand object categories [38].

[9] Image from Heuritech: Retrieved from https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/ on 1st August 2018.

5.2. RESNET18

Vanishing or exploding gradients (*while training a very deep network the derivatives can sometimes get extremely big or very small, which makes training difficult* [40]) hamper convergence or arriving at an optimal solution [41]. With deeper network depth accuracy gets saturated and then degrades rapidly [42].

The degradation problem is addressed by using a deep residual learning framework as shown in figure 5.2 [42].

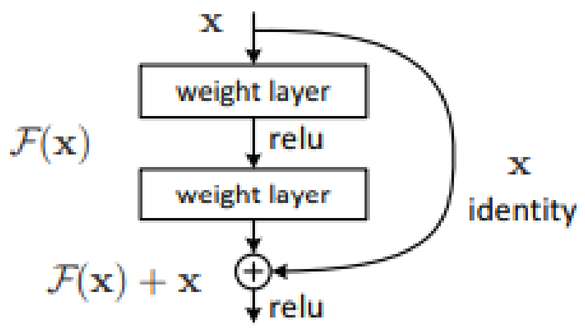
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig 5.2. (source :**Deep Residual Learning for Image Recognition** [42])[10]

Instead of having each few stacked layers fit an underlying desired mapping, the layers fit a residual mapping [42].

x denotes the input [42].

Let the desired underlying mapping be *H(x)* [42].

Let the stacked nonlinear layers fit another mapping of *F(x):=H(x)-x (assumption here is that the input and output are of the same dimensions)* [42].

The original mapping is recast into *F(x)+x* [42].

The shortcut connections performs the identity mapping, whose outputs are added to the output of the stacked layers [42].

[10] He K.,Zhang X.,Ren S.,Sun J.(2015),"Deep Residual Learning for Image Recognition" https://arxiv.org/pdf/1512.03385.pdf

Residual learning is applied to every few stacked layers [42].

A building block is defined as

$y=F(x,\{W_i\})+x,$

x and y are the input and output vectors of the layers [42]. $F(x,\{W_i\})$ represents the residual mapping to be learned [42]. Compared to VGG nets, ResNet has fewer filters and lower complexity [42].

Image resizing is done with the image's shorter side randomly sampled for scale augmentation [42]. Then a 224x224 crop of the image is randomly sampled with per-pixel mean subtracted [42].

An advantage of the ResNet architecture is that the gradients flow directly through the identity function from later layers to the earlier layers [43].

5.3. DENSENET

In this type of network, each layer is connected to every other layer in a feed-forward fashion. DenseNet has $\frac{L(L+1)}{2}$ direct connections, where L denotes the number of layers [43]. It has the following advantages: a) alleviates the vanishing-gradient problem b) strengthens feature propagation (*features extracted by very early layers are directly used by deep layers throughout the same dense block*) c) feature reusing capability d) reduction in the number of parameters [43].

The l^{th} layers has l inputs which consists of feature maps of preceding convolutional blocks [43]. Then its own feature-maps are passed on to all L- l subsequent layers, hence we get $\frac{L(L+1)}{2}$ connections in the L-layer network instead of just L as in traditional architectures [43]. Each of the layers implement a non-linear transformation $H_l(.)$ where l indexes the layer; x_l denotes the output of l^{th} layer; $x_l = H_l([x_0, x_1, \ldots, x_{l-1}])$. $[x_0, x_1, \ldots, x_{l-1}]$ are the feature maps produced in the layers 0 to l -1 [43].

Growth rate (k)

Each function H_l produces k feature maps [43]. The l^{th} layer has $k_0 + k \times (l - 1)$ input feature-maps; k_0 represents the number of channels in the input layer [43].

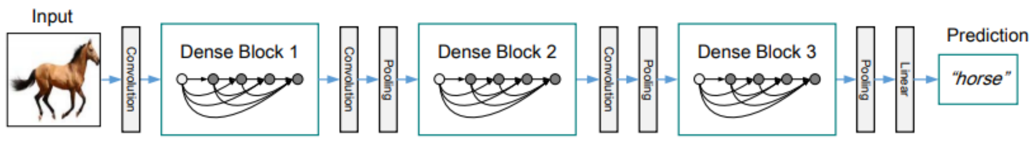

Fig. 5.3: A deep DENSENET with 3 dense blocks (source : **Densely connected convolutional networks** [43])[11]

In figure 5.3, *the layers between two adjacent blocks are referred to as transition layers which changes feature-map sizes via convolution and pooling* [43].

[11] Huang G.,Liu Z.,Maaten L.,Weinberger K.(2016) ,"Densely connected convolutional networks" *Conference on Computer Vision and Pattern Recognition 2017* https://arxiv.org/pdf/1608.06993.pdf

6. Defense against Adversarial Inputs

There is often disagreement among several models in predicting an input. This concept of disagreement among models is called prediction inconsistency [13]. So, by this theory an adversarial image may not fool every DNN model. Taking this idea into account an ensemble model is proposed involving some transformations such as image rotations and feature squeezing techniques on the input image which then is fed into an ensemble of pre-trained DNN models for predictions and the models' confidence in predicting the class of the object within the image is put through a majority vote.

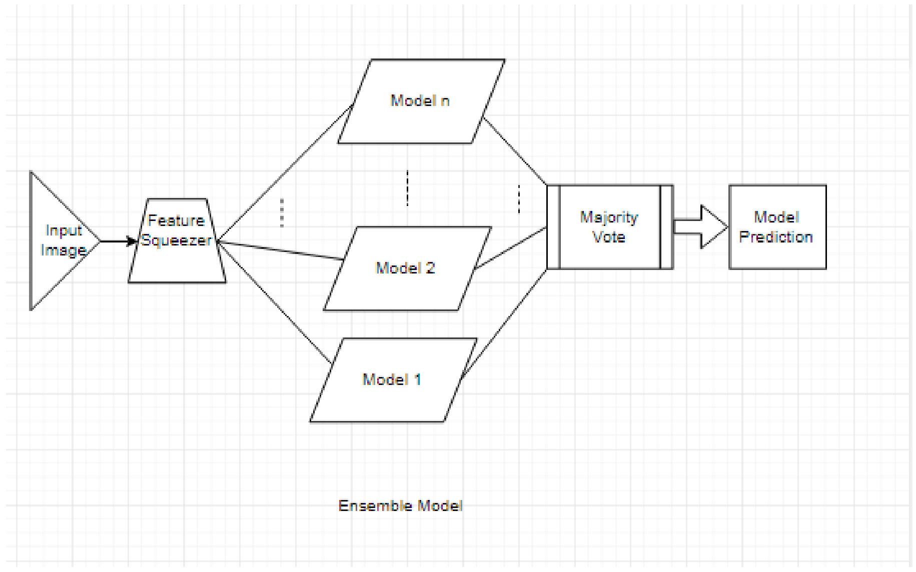

Fig. 6.1: shows a model with feature squeezer [13] and an ensemble of pre-trained image classifiers.

Description of the model:

The model as presented in Figure 6.1.takes in as input an image. The input image Q then goes through some feature squeezing technique, in the experiments concerning this thesis only median filtering has been dealt with, along with some transformation such as image rotations. The reason for doing this is explained further on in this chapter.

The transformed input, let us call it Q’, is then inputted into the sub-models, which are pre-trained deep neural networks (RESNET18, VGG16, DENSENET in our case). The number of pre-trained models should be odd. This is because in the next step a majority vote is applied based on the predictions by the respective models, and an odd population of models always gives a winner.

The model in Figure 6.1 should output an “adversarial warning” in case it detects an adversarial, and also when all the sub-models do not come to an agreement i.e. all the models give different predictions.

The model works as follows:

a. It takes in an image Q, which goes through the model without any input transformation such as feature squeezing or rotations, and the ensemble predicts its decision *f(x)=Q*, which is the class of the image.
b. Then the image Q is transformed into Q’ via input transformation techniques as discussed earlier and put through the various internal stages of the ensemble model. The model outputs its decision *f(x)=P.*
c. If P is equal to Q, the model’s output can be safely considered. If P≠Q ,the model gives off a warning that the input image is adversarial in nature.

The output of the ensemble model is the final prediction.

Here there are two lines of defense against the adversarial inputs:

- Input transformation to dilute the effect of adversarial examples.
- Majority voting which takes advantage of prediction inconsistency.

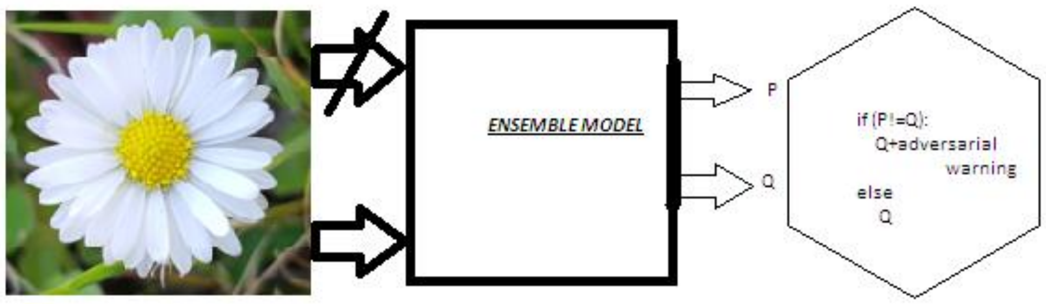

Fig. 6.2: illustrates the working of the model.

Figure 6.2 shows that an image is inputted into ensemble twice, once without input transformation (denoted by slashed arrow) and once with input transformation. For the two cases, the ensemble comes with predictions P and Q respectively.

If predictions P and Q are unequal the model gives its prediction Q of the transformed image with an adversarial warning. If both the predictions are equal the model just presents with the prediction Q of the transformed image.

Image transformation techniques:

- Feature Squeezing
- Image Rotations

<u>Feature Squeezing</u> [13]:

Most studies, such as adversarial training [3] and defense distillation [8], to defend against adversarial examples focused on transforming or refining the deep neural network models. Compared to refining or retraining DNNs feature squeezing is inexpensive and simple.

This approach is based on the observation that feature input space are unnecessarily large and provides ample opportunity for the creation of adversarial images [13]. The strategy is to squeeze the unnecessary input space, thereby limiting the space for adversarial inputs [13].

The paper [13] explores two simple squeezing techniques:

- Reduction of color-depth of each pixel in an image.
- Using spatial smoothing reduce the differences among individual pixels.

Color depth reduction:

A digital image is represented by an array of pixels, each of which is usually represented as a number that represents a specific color [13].

It is hypothesized that by reducing the bit depth, the adversarial opportunity itself gets reduced [13].

Spatial smoothing:

Spatial smoothing or blur is a technique of reducing noise in images [13].

There are two types of spatial smoothing: local and non-local [13].

In local smoothing each pixel uses its neighboring pixels to smooth itself; some of the common local smoothing methods are :

- Gaussian smoothing,
- Mean smoothing,
- Median smoothing [13].

Median filtering is the smoothing technique used for this thesis. It is also referred to as median blur or median filter [13].

Median filter uses a sliding window to move over each pixel of the image, and the central pixel is replaced by the median value of the neighboring pixels within the window [13].

Adversarial inputs are just numeric values within an array trained to fool a network.

To dilute the effect of these values, median filter is used. In doing so, the central values of the window gets replaced by the neighboring values of its pixels, hence removing the adversarial input to a certain extent.

Median filter is particularly effective at removing salt-and-pepper noise, which are sparsely occurring black and white pixels in an image [13].

Opencv's (see appendix) function 'medianBlur()' is used for median filtering with the window size being 3x3.

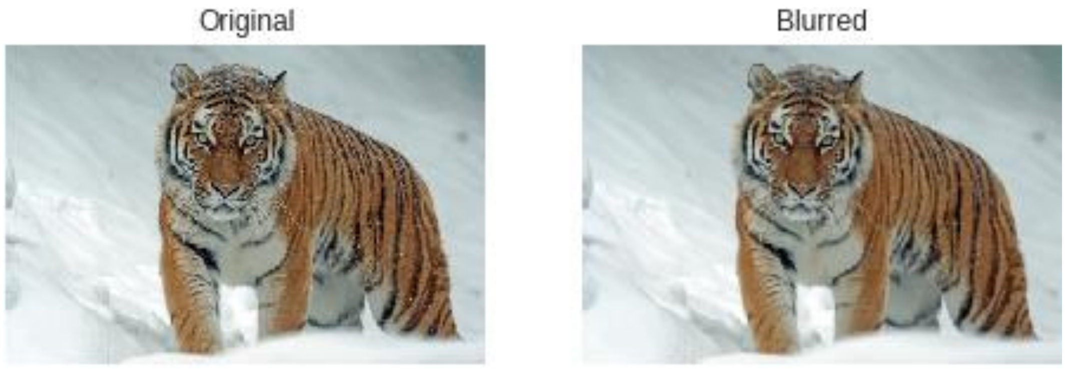

Fig.6.3. : Sliding window 3

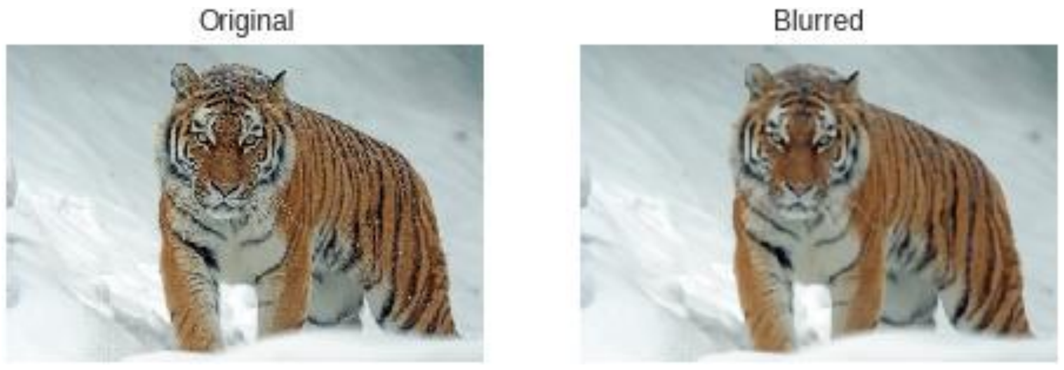

Fig. 6.4.: Sliding window 5

Figures 6.3 and 6.4 illustrates the effect of using various median filtering sliding windows on the quality of the image, the difference may not be much to the human eye, but to the deep neural network the difference may be drastic.

A sliding window of 3 is chosen so as to keep the distortions to an image as small as possible, as the blurring effect reduces the confidence of the model's predictions.

The table below shows the effect of the median blurring on the model's predictive ability. The model chosen is ResNet50. The table has 3 columns : first shows the size of the sliding window, if 0 it means no median filtering used; second and third columns show the top 2 predictions for the image by the model respectively.

Sliding Window (SW)	1st Prediction	2nd Prediction
SW=0	tiger : 68.9%	tiger_cat : 23.5%
SW=3	tiger : 63.7%	tiger_cat : 10.8%
SW=5	tiger : 27.7%	snow_leopard : 14.9%

Next, image in Fig. 6.5. was chosen whose confidence prediction by the model is low at 46.8%. This is done to test out the effect of blurring on low confidence images.

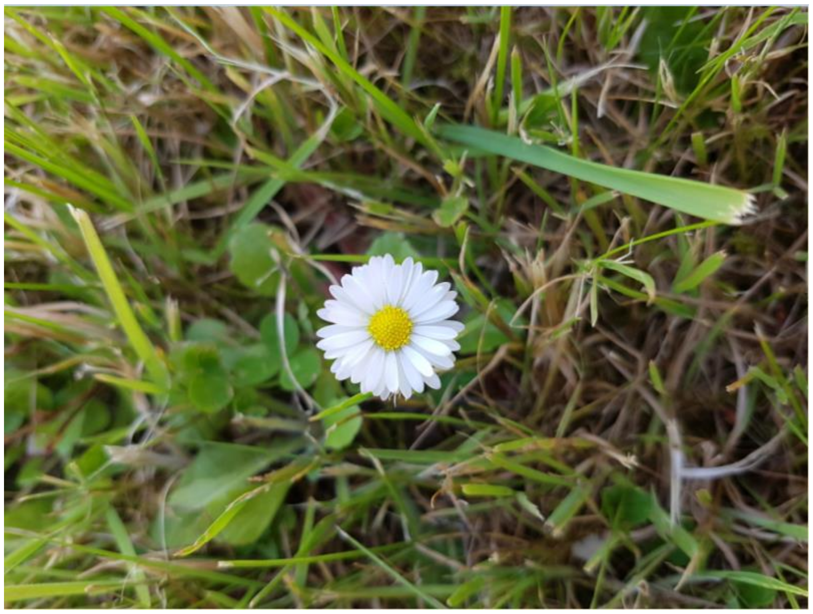

Fig. 6.5.

The table below shows the effect (on the model's prediction) upon using the sliding windows of sizes 0, 3, and 5.

Sliding Window (SW)	1st Prediction	2nd Prediction
SW=0	daisy : 46.8%	mushroom : 11.9%
SW=3	daisy : 51.1%	spaghetti_squash : 8.9%
SW=5	mushroom : 16.9%	stinkhorn : 12.3%

For both these images a worsening effect is seen on the model's classification ability upon using median filter of 5, with the case being even worse for the unperturbed images with low confidence.

In non-local smoothing, smoothing is done over similar pixels in a much wider area instead of neighboring pixels; a patch is selected, and the algorithm searches for similar patterns in the wide area of the image replacing the central patch with the mean of those patches [13].

<u>Image Rotations</u>:

Any upside-down image of an object may be easily recognized by an human eye, but to a DNN it is another image. A DNN reads an image as an array of numeric values, when presented with an upside-down image, the array values flips vertically.

Let the array A equals

[[1, 2]

 [3, 4]]

Upon flipping it upside down we get,

[[3, 4]

 [1, 2]]

"*Are neural network classifiers robust to simple, naturally-occurring transformations of their input?* [12]"

This question was asked in the paper [12], wherein it is shown that simple transformations which appear natural to humans may cause a significant drop in the model's performance.

Instead of generating adversarial examples, they (the authors of [12]) applied simple rotations and translations which proved quite effective in attacking state-of-the-art neural network models.

For this project, simple flipping of input images from left to right (mirror-image) is done to test out its effect on model's performance.

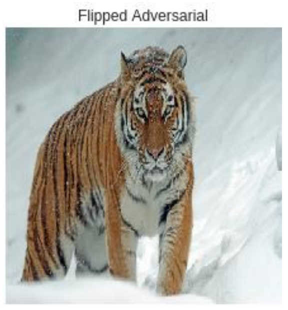

Fig. 6.6.

Adversarial image generated by targeting the image of the tiger (Fig. 6.3.) through the L-BFGS-B (chapter 4) is flipped horizontally. The below table shows the change in confidence upon rotating the images: both original unperturbed and adversarial. The first row shows the prediction of the model for the original image; second row for the original image flipped horizontally; third, for the adversarial; and fourth, for the adversarial flipped horizontally.

IMAGE (predicted class name)	**CONFIDENCE %**
Original (tiger)	68.9
Original (flipped:tiger)	40.6
Adversarial (airliner)	92.5
Adversarial (flipped:tiger)	93.1

Figure 6.6. shows the adversarial image flipped horizontally.

Another example showing the effect on another image (Fig. 6.5). The table below shows the results:

IMAGE (predicted class name)	**CONFIDENCE %**
Original (daisy)	46.8
Original (flipped:daisy)	68.9
Adversarial (airliner)	89.5
Adversarial (flipped:daisy)	97.9

For both these images, we see that the effect of adversarial images getting nullified, hence we may deduce that adversarial examples lose their robustness when subject to rotations. Adversarial examples are not rotation-invariant.

Later on we'll test this conjecture on a dataset of more images.

Smoothing and rotation together as a defense:

Earlier we saw that upon simple rotation of the image, in our case through horizontal flip, and median filtering the adversarial images lost its robustness. The adversarial inputs no longer fooled the system it was trying to deceive.

The next approach was to use both these techniques of defense together and see the results.

First, we'll see the effect of applying image rotation and then median smoothing on the images of tiger and daisy, as we have done before.

The table below shows the data related to the RESNET50's predictions when these transformations are put into place once on the original, unperturbed image and then on the adversarial image. The first row shows the effect on the model upon applying the defenses on the original image of a daisy; second, on the adversarial image of a daisy initially labeled as an airliner; third, the defenses upon the original image of a tiger; and fourth, on the adversarial image of a tiger.

IMAGE (median filter + horizontal flip)	**CONFIDENCE %**	**PREDICTED CLASS**
Original:daisy	78.9	Daisy
Adversarial:airliner	97.2	Daisy
Original:tiger	40.3	Tiger
Adversarial:airliner	91.1	Tiger

We see that adversarial inputs lose its robustness when subjected to both these types of simple defenses; and all the images whether adversarial or not are classified correctly.

Albeit the concern here was the model’s performance on the original images which had the defenses applied upon it. We see for daisy, there is a sharp increase in the model’s performance towards predicting its class correctly, and also an opposite decline in its ability to correctly classify the image of a tiger. Even though it correctly classified both these images after transformations the fluctuations in model’s performance shows that it is quite unstable in its predictions.

A brief recapitulation about the structure of the ensemble model used:

No. of pre-trained models used: ***3 (RESNET18, VGG16, DENSENET)***

Initial input transformations: ***rotation + median filtering***

The attack (creating a new attack system):

The following is how the model in figure 6.1. is attacked to test its performance.

First, let us state the assumptions:

- Attacker knows two out of the three pre-trained models used.(RESNET18, VGG16 in our experiments)
- The attacker is unaware of the third model. (DENSENET)
- The experiments are done in such a way that the attacker tries to fool both of these models by generating adversarial examples for both with high fooling confidence (95% in our experiments).
  A brief introduction to the attack method:
  Let the original image be Q.
  The adversarial for one model be Q’ and another be Q’’.
  The adversarial input (just the perturbation) is then extracted from both the adversarial examples by simple subtraction of arrays i.e. original array of Q subtracted once from Q’ and then from Q’’.
  $P'=Q'-Q$
  $P''=Q''-Q$

Then we add these adversarial perturbations to the original image Q to generate a new image α=Q+P'+P''

Now these additions may exceed the initial boundary limitations and may no longer represent an image array, so any element of α exceeding the lower and upper bounds of the original image are clipped off to its nearest limit. For example if the limits are 0 and 1; values -1.2 and 2.1 would be clipped off to 0 and 1.

We call this final new adversarial image α'.

- α' is then inputted into the model, and it goes through the ensemble as described in Figure 6.2. It is to be noted that the attacker creates adversaries of the same class.

Rotation a failed attempt:

The experiment was done initially with 60 different images downloaded from the ImageNet website.

First the original image was put into the model and then the adversarial α'.

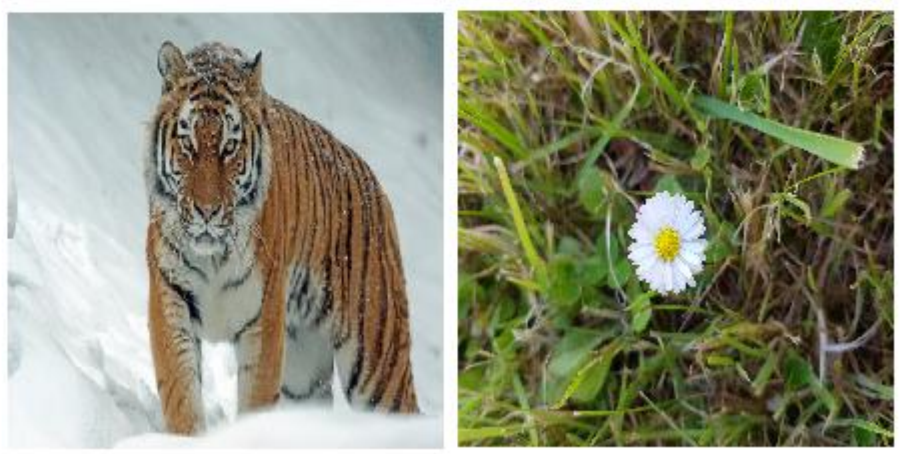

Fig. 6.7 : α' for Tiger and Daisy.

Figure 6.7 shows the α' s for the two example images (tiger and daisy). As we can see the added perturbations is imperceptible to the human eye.

The effect of the transformed original image was noted; secondly, it was checked if α’ is indeed of the targeted adversarial class, lastly if the defense is working i.e. if it correctly classified the input image by bypassing the adversarial inputs.

Following are the results which we got (as shown in the table below).The first column shows how good the defense was in correctly classifying the original unperturbed images. The second column shows the percentage of success in creating α’. There is a chance that α’ may not be robust due to the clipping of values crossing the boundary limits and also the effect gets somewhat thwarted due to the triad models used as the second line of defense. The third column just shows how successful the ensemble was in defending against adversarial images.

ORIGINAL IMAGE SUCCESS RATE %	**SUCCESSFUL CREATION OF α’ %**	**SUCCESS OF DEFENCE %**
47.6	62	44.4

The model just was successful in defending in 44.4% of the cases, and performed awfully with just 47.6% even when the input was not adversarial i.e. the model was unable to predict correctly the images (unperturbed) with rotations as input transformations.Another way of putting it is, the dual defense method applied on the original images led to it be misclassified by the model.

Although we saw that the attacked images were successfully fooling the model 62% of the times without the first line of defense (the image transformation).

Keeping this failed attempt in mind, following changes are made to the model:

No. of pre-trained models used: ***3 (RESNET18, VGG16, DENSENET)***

Initial input transformations: ***median filtering***

As was suspected earlier the rotations aren’t a reliable source of defense, as it caused even the unperturbed images to get misclassified, this claim is established by dropping horizontal flipping of images as a defense.

The ensemble model now has a simple median filtering as input transformation, and 3 pre-trained models as second line of defense.

The table below gives the information related to how well the new ensemble works.

Now, the experiments were conducted with 120 different images.

ORIGINAL IMAGE SUCCESS RATE %	**SUCCESSFUL CREATION OF α' %**	**SUCCESS OF DEFENCE %**
90	65.8	87.5

We see that 87.5% of the times the model correctly classified or defended against adversarial examples. This 12.5% misclassification rate may be due to the fact that the median filtering made 10% of the original, unperturbed images get misclassified. This 10% value clearly shows that 3 x 3median filtering is a better defense technique compared to horizontal flipping of images. When we compared the accuracy with the class outputted by the model just on the unperturbed, original images we got an accuracy of 97.5%.

It is seen that 65.8% of the images fooled the triad:pre-trained models (considering the majority vote) without the first line of defense (median filtering).

Next, out of the 65.8% of the times the ensemble (without the filtering effect) was fooled, we see that the model was 82.3% of times successful in predicting correctly the class of the input image. Again, 17.7% of the times the model was unsuccessful only in correctly predicting the class of the image. But the percentage of success is relatively quite high.

Note: All the pre-trained models used in the ensemble had the input images normalized in the following way: mini-batches of 3-channel (RGB) images of shape 3x224x224; the images were then loaded into a range of [0-1] and then normalized using mean =[0.485,0.456,0.406] and std=[0.229,0.224,0.225].

7. Conclusion

The statement “Computer vision using deep neural network has surpassed human visual system” has often been used in describing the success stories related to deep learning. However, the images supplied as data for training are labeled by humans themselves, so the networks cannot be said to have exceeded the level of superiority of humans.

To give an example why it is true that Computer vision has still a long way to go, we demonstrate it using the figure below.

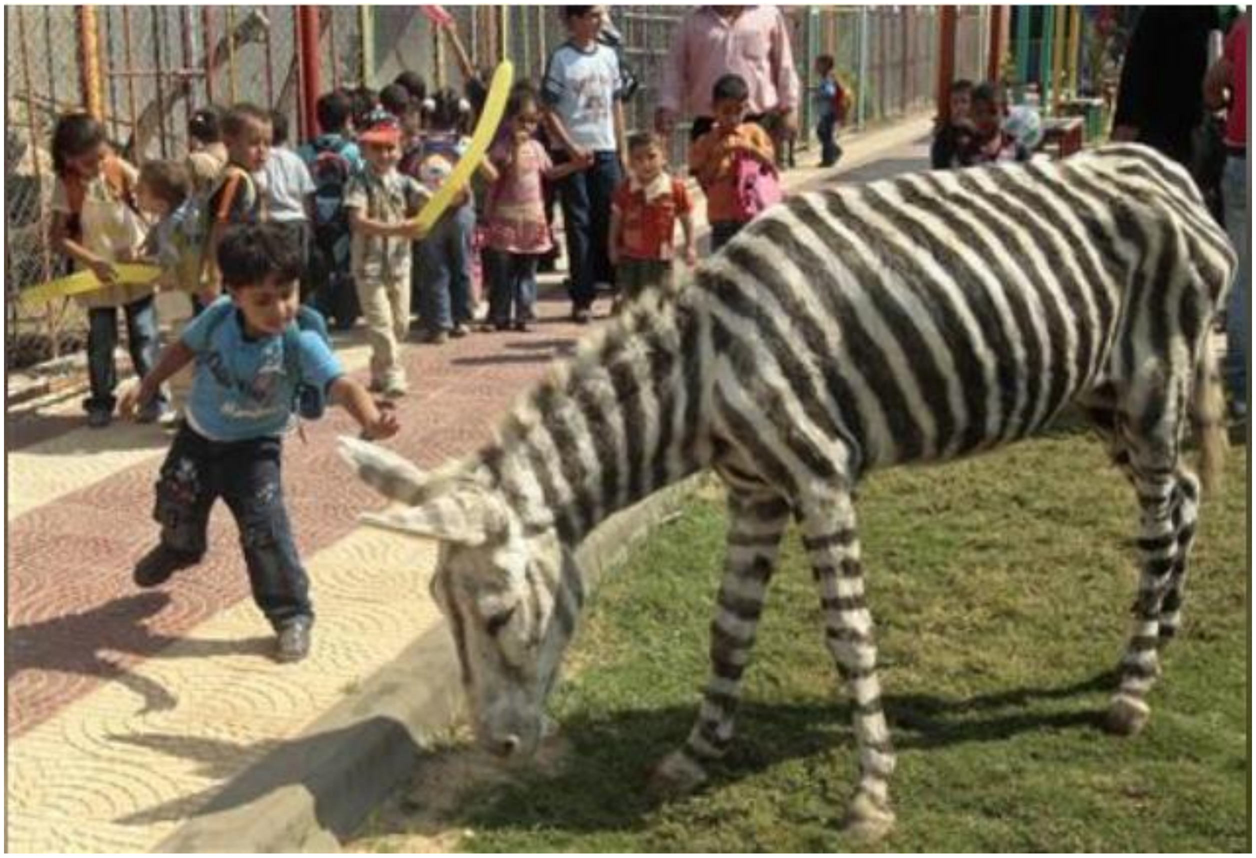

Fig.7.1: Donkey painted to look like Zebra (source : **reuters.com** [44])[12]

[12] Image source. Retrieved from https://www.reuters.com/article/us-gaza-zebras/donkeys-get-dye-job-take-on-zebra-role-idUSTRE5973NV20091008 on 19th August 2018.

Figure 7.1 shows an image of a donkey painted by a zoo to look like a zebra. Most humans who'd just glance at it may call it zebra, but upon careful observation the deception at work here may be caught.

This particular image when fed into a pre-trained DENSENET model gets classified by it as a 'zebra'. The image was not an adversarial image, but it nevertheless fooled the model.

A model therefore is as good as the training data it receives, and every model known today is still dependent upon the humans for annotation.

In this thesis we studied object detection using YOLO object detection system, and dealt with a technique to defend against adversarial images.

We saw how the YOLO object detection system is faster than most region-based object detection systems, but falls behind them in terms of accuracy. The speed of YOLO makes it a good candidate for autonomous vehicles, where the detection system must work fast.

The second part of the thesis dealt with adversarial examples. We used L-BFGS-B to generate adversarial examples. Using this technique we created a new attack method, where instead of fooling just one of the pre-trained models, the attack system was able to fool two of the models about 65% of the time. It was shown how flipping images horizontally fools the network without it meaning to be adversarial in nature. Here also, we see the flaw in the deep neural network. The training data should also be in various rotated and translated formats. Relative to rotations we demonstrated that smoothing techniques are a better defense option. Our ensemble was able to not just detect adversarial images but also able to defend them 87.5% of the time. We were able to use simple input modification technique with an ensemble to defend against adversarial images. This method is much cheaper, and simple relative to the ones involving retraining a network.

8. Areas of improvement

YOLO object detection though faster than most state-of-the-art object detection systems, remains behind many in terms of accuracy. It fails to detect smaller objects within an image.

The second half of the thesis deals with the defense against adversarial examples. It had two parts of defense: the first being input or image modifications, and second being the ensemble of pre-trained models. Only two input modifications have been dealt with in this project: image rotation and median filtering; out of which, we demonstrated that horizontal flipping of an image is quite a bad defense. Other smoothing techniques such as gaussian filtering, bilateral filtering should also be used to check its implications on the networks' confidence. Another image modification technique called colour-depth reduction should also be used to see its impact on the model's prediction abilities.

Only L-BFGS-B technique was used to generate adversarial images, there are many other adversarial image-generating algorithms which could be used.

The ensemble model was tested on 120 images of 40 categories in total. It is yet to be tested on a wide number and variety of images.

9. References

[1] Fridman L.(2018),"Deep learning for self-driving cars", MIT. Retrieved from https://selfdrivingcars.mit.edu on 21st July 2018.

[2] Huan T.(1996), "Computer vision:Evolution and Promise." 5th *International Conference ,High technology: Imaging science and technology.* https://cds.cern.ch/record/400313/files/p21.pdf

[3]Goodfellow I.,Shlens J.,Szegedy (2014) ," Explaining and Harnessing Adversarial examples". *International Conference on Learning Representations 2015.* https://arxiv.org/pdf/1412.6572.pdf

[4]Tuzel O.,Shrivastava A.,Moosavi-Dezfooli S.(2018)," Divide, Denoise, and Defend against Adversarial Attacks." https://arxiv.org/pdf/1802.06806.pdf

[5] ImageNet . Retrieved from http://www.image-net.org/about-overview on 21st July 2018.

[6] Goodfellow I.,Papernot N.,Huang S.,Duan Y.,Abbeel P.,Clark J. (2017). Retrieved from https://blog.openai.com/adversarial-example-research/ on 21st July 2018.

[7]Hinton G.,Vinyals O.,Dean J.(2015)," Distilling the knowledge in a neural network." *Conference on Neural Information Processing Systems 2014 Deep Learning Workshop* https://arxiv.org/pdf/1503.02531.pdf

[8]Papernot N., McDaniel P., Wu X., Jha S. (2015),"Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks." *IEEE Symposium on Security & Privacy 2016.* https://arxiv.org/pdf/1511.04508.pdf

[9]Carlini N.,Wagner D.(2016)," Defense Distillation is Not Robust to Adversarial Examples." https://arxiv.org/pdf/1607.04311.pdf

[10]Carlini N., Wagner D.(2017)," Adversarial Examples are not easily detected: Bypassing ten detection methods." https://arxiv.org/pdf/1705.07263.pdf

[11] YOLO. Retrieved from https://pjreddie.com/darknet/yolo/ on 21st July 2018.

[12]Tran B.,Engstrom L.,Tsipras D.,Schimdt L.,Madry A.(2017)," A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations." *Neural Information Processing Systems Workshop on Machine Learning and Computer Security*. https://arxiv.org/pdf/1712.02779.pdf

[13]Xu W.,Evans D.,Qi Y.(2017)," Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks." *Network and Distributed Systems Security Symposium 2018.* https://arxiv.org/pdf/1704.01155.pdf

[14]Athalye A. ,Engstrom L.,Ilyas A.,Kwok K.(2017),"Synthesizing Robust Adversarial Examples." *International Conference on Machine Learning 2018.* https://arxiv.org/pdf/1707.07397.pdf

[15]Goodfellow I.,Bengio Y.,Courville A.(2016)," Chapter 6 : Deep Feedforward Networks" https://www.deeplearningbook.org/contents/mlp.html

[16]Goodfellow I.,Bengio Y.,Courville A.(2016), "Part II : Deep Networks : Modern Practices " https://www.deeplearningbook.org/contents/part_practical.html

[17]Ciresan D.,Meier U., Schimdhuber J.(2012), "Multi-column Deep Neural Networks for Image Classification" *Dalle Molle Institute for Artificial Intelligence.* https://arxiv.org/pdf/1202.2745.pdf

[18] The British Machine Vision Association and Society for Pattern Recognition, "What is computer vision" http://www.bmva.org/visionoverview Retrieved on 24th August 2018.

[19]Goodfellow I.,Bengio Y.,Courville A.(2016), "Chapter 9 : Convolutional Networks" https://www.deeplearningbook.org/contents/convnets.html

[20] Chris Urmson.," The google self-driving car project." Talk given at Robotics: Science and Systems, 2011.

[21]The Guardian (19 March 2018),"Self-driving Uber kills Arizona woman in first fatal crash involving pedestrian" https://www.theguardian.com/technology/2018/mar/19/uber-self-driving-car-kills-woman-arizona-tempe

[22]Teichman A.,Thrun S. (2011)," Practical object recognition in autonomous driving and beyond" *Advanced Robotics and its Social Impacts 2011* https://ieeexplore.ieee.org/document/6301978/

[23]Szegedy C.,Zaremba W., Sutskever I.,Bruna J.,Erhan D., Goodfellow I.,Fergus R.(2013),"Intriguing properties of neural networks" https://arxiv.org/abs/1312.6199

[24]Rauber J.,Brendel W.,Bethge M.(2017), "Foolbox: A Python toolbox to benchmark the robustness of machine learning models" *International Conference on Machine Learning 2017* https://arxiv.org/pdf/1707.04131.pdf

[25]Datacamp. Retrieved from https://www.datacamp.com/community/tutorials/object-detection-guide on 10th August 2018.

[26]Redmon J.,Divvala S.,Girshick R.,Farhadi A.(2015)," You Only Look Once: Unified, Real-Time Object Detection" https://arxiv.org/pdf/1506.02640.pdf

[27]Madden M. (2018), "Lectures: Advanced Topics in Machine Learning & Information Retrieval", National University of Ireland, Galway

[28]Girshick R.(2015),"Fast R-CNN" *International Conference on Computer Vision 2015* https://arxiv.org/pdf/1504.08083.pdf

[29]Mona lisa ,Louvre. Retrieved from https://www.louvre.fr/en/mediaimages/portrait-de-lisa-gherardini-epouse-de-francesco-del-giocondo-dite-monna-lisa-la-gioconda on 11th August 2018.

[30]Darkflow. Retrieved from https://github.com/thtrieu/darkflow on 30th June 2018.

[31]CUDA. https://en.wikipedia.org/wiki/CUDA Retrieved on 25th August 2018.

[32]Darknet. Retrieved from https://pjreddie.com/darknet/yolov2/ on 21st July 2018.

[33]Road signs . Retrieved from http://www.epermittest.com/road-signs on 1st August 2018.

[34]Tabacof P., Valle E.(2016), "Exploring the space of Adversarial Examples" *IEEE 2016* https://arxiv.org/pdf/1510.05328.pdf

[35]Rasin I., Pumperla M., Hanlon T., Pansare N., Kienzler R., "Applied AI with Deep Learning" Coursera course. https://www.coursera.org/learn/ai

[36] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal (1994),"L-bfgsb: Fortran subroutines for large-scale bound-constrained optimization" https://www.semanticscholar.org/paper/NORTHWESTERN-UNIVERSITY-Department-of-Electrical-L-Zhu/5a2253e72c71012c3a970689c1627492172440b7?tab=references

[37] Image. Retrieved from https://www.pexels.com/search/tiger/ on 21st July 2018.

[38] Simonyan K., Zisserman A.(2014), "Very Deep Convolutional Networks for Large-Scale Image Recognition." *International Conference on Learning Representations 2015* https://arxiv.org/pdf/1409.1556.pdf

[39]Image from Heuritech: Retrieved from https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/ on 1st August 2018.

[40]Ng A., "Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization" https://www.coursera.org/lecture/deep-neural-network/vanishing-exploding-gradients-C9iQO

[41] Y. Bengio, P. Simard, and P. Frasconi.(1994)," Learning long-term dependencies with gradient descent is difficult" http://www.iro.umontreal.ca/~lisa/pointeurs/ieeetrnn94.pdf

[42]He K.,Zhang X.,Ren S.,Sun J.(2015),"Deep Residual Learning for Image Recognition" https://arxiv.org/pdf/1512.03385.pdf

[43]Huang G.,Liu Z.,Maaten L.,Weinberger K.(2016) ,"Densely connected convolutional networks" *Conference on Computer Vision and Pattern Recognition 2017* https://arxiv.org/pdf/1608.06993.pdf

[44]Image source. Retrieved from https://www.reuters.com/article/us-gaza-zebras/donkeys-get-dye-job-take-on-zebra-role-idUSTRE5973NV20091008 on 19th August 2018.

10. Appendix

Tools used in this project.

Google Colab : *free jupyter notebook environment, runs entirely in the cloud.* https://colab.research.google.com/

Python : *programming language* https://www.python.org/

Foolbox : *python toolbox to create adversarial examples that fool neural networks.* https://foolbox.readthedocs.io/en/latest/

Torchvision : *package consists of popular datasets, model architectures, and common image transformations for computer vision.* https://pytorch.org/docs/stable/torchvision/index.html

Darkflow : *darknet (open source neural networks in C) translated to Tensorflow.* https://github.com/thtrieu/darkflow

Tensorflow : *open source software library for high performance numerical computation.* https://www.tensorflow.org/

Keras : *high-level neural networks API, written in Python.* https://keras.io/

OpenCV : *open source computer vision library* https://opencv.org/

COCO : *large-scale object detection, segmentation, and captioning dataset.* http://cocodataset.org/#home

PyTorch : *python package that provides tensor computation like numpy and deep neural networks.* https://pytorch.org/about/

Some parts from the code (Defense against Adversarial Images):

```
# ## First line of defense

# In[9]:


import numpy
import cv2
def defense(altered):#median filter using 3 x 3 filter
  return cv2.medianBlur(altered,3)


# ## Second line of defense

# In[10]:


def ensemble(inputImage):#Must be odd Numbers
  listPredictions=[]#list would contain the predictions
  listPredictions.append(np.argmax(fmodel.predictions(inputImage)))
  listPredictions.append(np.argmax(fmodel1.predictions(inputImage)))
  listPredictions.append(np.argmax(fmodel2.predictions(inputImage)))


  if(listPredictions.count(max(set(listPredictions),key=listPredictions.count
))==1):#if the maximum class occurs just once
    return "***adversarial***"#in case all models predict different classes
  else:
    return
max(set(listPredictions),key=listPredictions.count),listPredictions#sends the
class with maximum occurence


# ## The complete model including both the defenses

# In[11]:


def defenseEnsemble(inputImage):#uses ensemble(abc) and defense(xyz)
functions
  x=ensemble(inputImage)
  y=ensemble(defense(inputImage))
  print("Prediction Before Defense",x)
  print("Prediction After Defense",y)
  if(x!=y):
    print("Warning:Adversarial Input")
  return x[0],y[0]#returns the max predictions
```

```
# ## Attack and the defense

# In[13]:


 image,fmodel,adversarial=RESNET18(390)#would classify as eel 390(ATTACKED)*
 image1,fmodel1,adversarial1=VGG16(390)#would classify as eel 390(ATTACKED)*
 image2,fmodel2=DENSENET()#unattacked DENSENET model(UNATTACKED)*

 #Differences
 diffVGG=adversarial1-image1
 diffRES=adversarial-image

 perturbation=image+diffRES+diffVGG#addition of the noise to the input image
 perturbation=numpy.clip(perturbation,0,1)#clipping so that the boundary
conditions are not violated. #Attacked Image

 defenseEnsemble(image) #Input : original Image

 defenseEnsemble(perturbation) #Input : Adversarial Image
```

* ***These are the functions (see GitHub link in the next page) which returns the images (arrays), unperturbed and adversarial (if attacked), along with their respective models.***

*Full code written and executed as a part of this project is available in a CD attached with the hard-copy of the thesis. Code has also been uploaded in **GitHub** link given below.*

GitHub: https://github.com/ashfaqueazad/MScThesis-Code2018

Contained within these are:

- Code for object detection using YOLO v2 608 x 608.
- Defense system :
  - Defense model against adversarial attacks.
    - Also calculates the accuracy of the Defense model.
  - Code showing the creation of α' attack.*
  - Using L-BFGS-B to attack ResNet50 and generate adversarial image.
    - Also checks the impact of using rotation (in our case flipping an image horizontally) and median filter (with window size of 3 x 3) upon images, both adversarial and unperturbed.

* α' is an adversarial image capable of attacking two pre-trained image classifiers at a single time.

****