Deep Learning: Homework2

1706 Wanru Zhao(17373240)

April 2020

Most existing machine learning classifiers are highly vulnerable to adversarial examples. An adversarial example is a sample of input data which has been modified very slightly in a way that is intended to cause a machine learning classifier to misclassify it. In many cases, these modifications can be so subtle that a human observer does not even notice the modification at all, yet the classifier still makes a mistake.

Adversarial examples pose security concerns because they could be used to perform an attack on machine learning systems, even if the adversary has no access to the underlying model.

1 Generating untargeted adversarial examples

The goal of the non-targeted attack is to slightly modify source image in a way that image will be classified incorrectly by generally unknown machine learning classifier.

The goal of the targeted attack is to slightly modify source image in a way that image will be classified as specified target class by generally unknown machine learning classifier.

1.1 Method

Run the baseline code and record the perturbation scale and success rate.

1.2 Results

Result of Generating untargeted adversarial examples	
Success rate of this attack	0.98
Noise norm of this attack	59.122650146484375

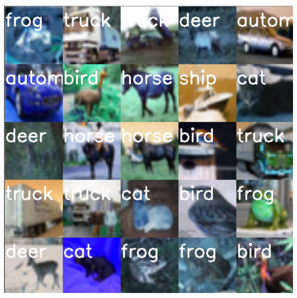

Figure 1: Baseline

2 Adding loss terms

1 Adding averaged total variation loss and see how the perturbation scale and success rate changes.

2 To obtain minimum distortion, add l2 loss between adversarial examples and the original images. Rerun the experiment to see how success rate changes with different c value, e.g. 1, 0.1, 0.01 etc.

2.1 Method

Total variation is defined as $\sum_{i,j}(y_{i+1,j} - y_{i,j})^2 + (y_{i,j+1} - y_{i,j})^2$ where i,j denotes row and column index of a pixel in the image. Total variation loss encourages images to be made up of piece-wise constant patches.

```
tv_loss = tf.reduce_mean(tf.image.total_variation(x_norm))
loss +=  tv_loss
```

L2 loss is defined as follow.

```
c = 0.001
l2_loss = tf.reduce_mean((x_round - x) ** 2)
loss += l2_loss * c
```

2.2 Results

Loss type	Value of C	Success rate	Noise norm
tv loss	1	0.98	52.74232482910156
l2 loss	1	0.92	0.7743880748748779
l2 loss	0.1	0.92	7.215312480926514
l2 loss	0.01	0.96	41.2048454284668
l2 loss	0.001	0.98	41.43083190917969

2.3 Analysis

- TV loss smoothes the images, thereby limiting the noise distribution of adversarial samples. So, we can concluded that TV loss can improve the robustness of samples and improve the defend adversarial attacks.
- L2 regularization between the original image and the adversarial examples means that the gap between the original image and adversarial examples should not be too large, to be specific, the nosie can not be too large. We should add as little noise as possible in case of misclassification. In fact, this operation is also a adversatial process, that is, to ensure the adversatial samples are misclassified as much as possible and the sample and the original image are as close as possible.
- The larger the value of c, the larger the success rate as well as the noise norm.

3 Whether augmentation helps defending adversarial attacks?

Implement one or some of the augmentation techniques, e.g. affine transformation, adding salt and pepper noise, bluring etc. on the generated adversarial examples. Evaluate the augmented adversarial examples and find out whether these examples still get misclassified.

3.1 Method

1 Affine transformation

```
def affineTrans(img):
    pts1 = np.float32([[10, 10], [20, 5], [5, 20]])
    pts2 = np.float32([[10, 8], [18, 5], [5, 20]])
    M = cv2.getAffineTransform(pts1, pts2)
    return cv2.warpAffine(img, M, (32, 32))
```

2 Salt and pepper noise

```
def noise(img, SNR=0.7):
    img_ = img.transpose(2, 1, 0)
    c, h, w = img_.shape
    mask = np.random.choice((0, 1, 2), size=(1, h, w),
                p=[SNR, (1 - SNR) / 2., (1 - SNR) / 2.])
    mask = np.repeat(mask, c, axis=0)
    img_[mask == 1] = 255  # pepper
    img_[mask == 2] = 0  # white
    return img_.transpose(2, 1, 0)
```

3 Bluring

```
    img = cv2.blur(img, kernel = (1, 1))
```

4 Blending of all 3 methods

```
+             img = affineTrans(img)
+             img = cv2.blur(img, (5, 5))
+             img = noise(img, 0.5)
```

3.2 Results

Method	Success rate	Noise norm
Affine transformation	0.96	25.645957946777344
Salt and pepper noise	0.96	24.998855590820312
Bluring	0.98	25.155427932739258
Blending of all 3 methods	1.0	25.63494873046875

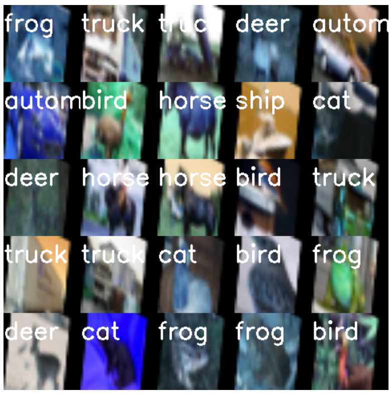
frog truck truc deer autom
autom bird horse ship cat
deer horse horse bird truck
truck truck cat bird frog
deer cat frog frog bird

Figure 2: Affine transformation

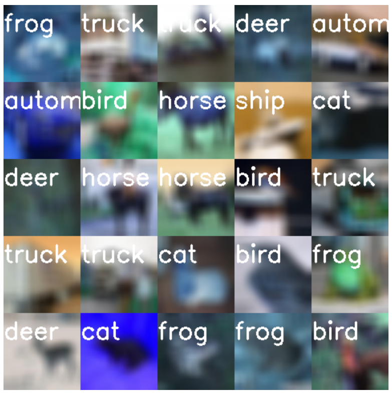
frog truck ruck deer autom
autom bird horse ship cat
deer horse horse bird truck
truck truck cat bird frog
deer cat frog frog bird

Figure 3: Bluring

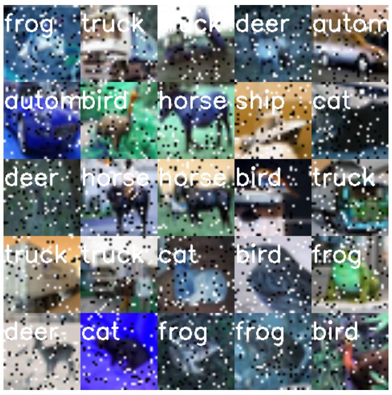
frog truck ck deer autom
autom bird horse ship cat
deer horse horse bird truck
truck truck cat bird frog
deer cat frog frog bird

Figure 4: Salt and pepper noise

3.3 Analysis

It can be seen from the experimental results that the blending of all 3 methods reaches the best performance. Each three methods has similar success rate and noise norm.

4 Generating targeted adversarial examples

In targeted attack scenario, please assign a target label for each image. After training, the corresponding examples will be classified as this label. You need to modify the loss term and rerun the generating process. You may also increase the number of epochs in each run.

4.1 Method

We target all imagesto be classified into class 0, that is, truck.

```
        loss = tf.reduce_sum(tf.nn.softmax_cross_entropy_with_logits
                                        (labels=target_one_hot, logits=logits))
```

```
        target = np.array([7])
```

```
-       placeholders['label']: labels,
+       placeholders['label']: target,
```

4.2 Results

Generating targeted adversarial examples	
Success rate	0.89
Noise norm	63.144779205322266

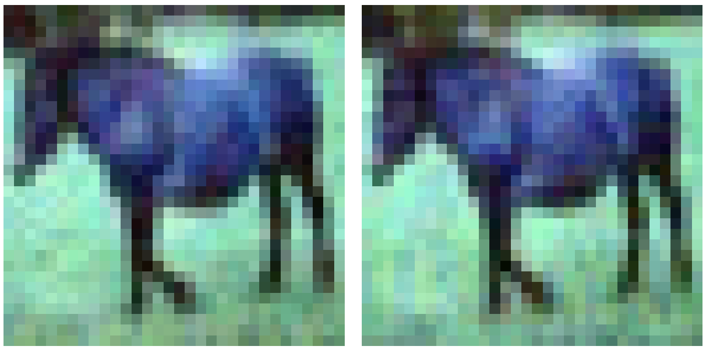
(a) Before: horse (b) After: truck

Figure 5: Original figure and adversarial figure

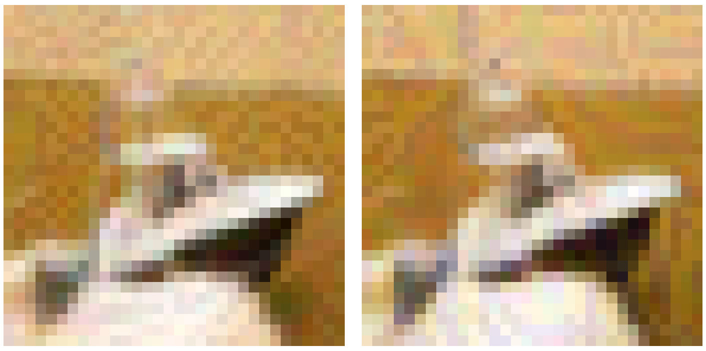
(a) Before: ship (b) After: truck

Figure 6: Original figure and adversarial figure

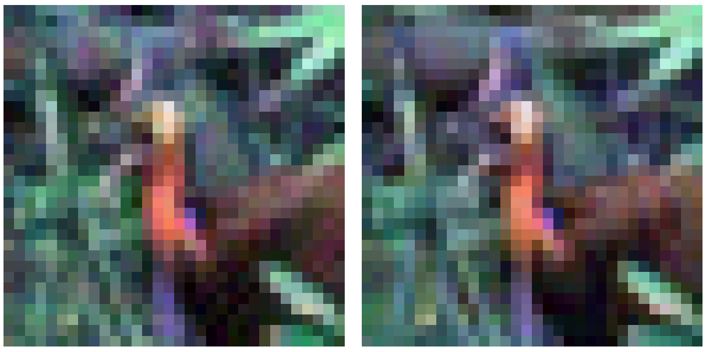
(a) Before: bird (b) After: trunk

Figure 7: Original figure and adversarial figure

4.3 Analysis

- It can be seen from the experimental results that targeted adversarial performs worse than untargeted adversarial attack on success rate, which stands for the probability of generated adversarial examples being misclassified by the model.