



adversarial-machine-learning

https://www.tensorflow.org/tutorials/generative/adversarial_fgsm

What is an adversarial example?




1. Adversarial examples are specialised inputs created with the purpose of confusing a neural network, resulting in the misclassification of a given input.
2. These notorious inputs are indistinguishable to the human eye, but cause the network to fail to identify the contents of the image.
3. There are several types of such attacks, however, here the focus is on the fast gradient sign method attack, which is a white box attack whose goal is to ensure misclassification.
4. A white box attack is where the attacker has complete access to the model being attacked.



What is an adversarial example?

Example:

Here, starting with the image of a panda, the attacker adds small perturbations (distortions) to the original image, which results in the model labelling this image as a gibbon, with high confidence. The process of adding these perturbations is explained below.

	$+ .007 \times$		$=$	
x		$\text{sign}(\nabla_x J(\theta, x, y))$		$x + \epsilon \text{sign}(\nabla_x J(\theta, x, y))$
“panda”		“nematode”		“gibbon”
57.7% confidence		8.2% confidence		99.3 % confidence

Fast gradient sign method

The fast gradient sign method works by using the gradients of the neural network to create an adversarial example.

For an input image, the method uses the gradients of the loss with respect to the input image to create a new image that maximises the loss. This new image is called the adversarial image.

$$adv_x = x + \epsilon * \text{sign}(\nabla_x J(\theta, x, y))$$

- adv_x : Adversarial image.
- x : Original input image.
- y : Original input label.
- ϵ : Multiplier to ensure the perturbations are small.
- θ : Model parameters.
- J : Loss.

Implementing fast gradient sign method

The first step is to create perturbations.

```
loss_object = tf.keras.losses.CategoricalCrossentropy()

def create_adversarial_pattern(input_image, input_label):
    input_label = label
    with tf.GradientTape() as tape:
        tape.watch(input_image)
        prediction = pretrained_model(input_image)
        loss = loss_object(input_label, prediction)

    # Get the gradients of the loss w.r.t to the input image.
    gradient = tape.gradient(loss, input_image)

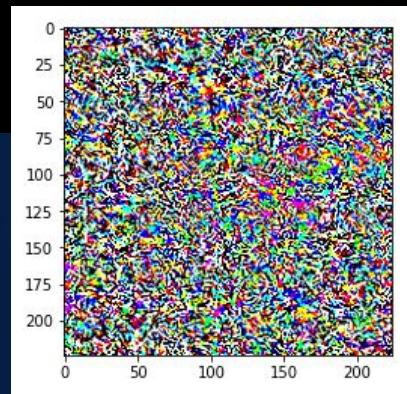
    # Get the sign of the gradients to create the perturbation
    return tf.sign(gradient)
```

Implementing fast gradient sign method

```
labrador_retriever_index = top_inds[0]
label = tf.one_hot(labrador_retriever_index, image_probs.shape[-1])
label = tf.reshape(label, (1, image_probs.shape[-1]))
input_image = tf.convert_to_tensor(x, dtype=tf.float32)

perturbations = create_adversarial_pattern(input_image, label)
plt.imshow(perturbations[0]*0.5+0.5);
plt.plot()
```

The resulting perturbations can also be visualised.



Different Values of Epsilon

Try different values of epsilon and see the results.

```
epsilons = [0, 0.5, 1, 20, 50]

for i, eps in enumerate(epsilons):
    adv_x = input_image + eps*perturbations
    pred = net.predict(adv_x, steps=10)[0]
    top_inds = pred.argsort()[::-1][:5]
    print('adversarial_' + f, ' is ', cls_list[top_inds[0]])
    for i in top_inds:
        print('    {:.3f}  {}'.format(pred[i], cls_list[i]))
    adv_x = preprocess(adv_x)
    adv_x = tf.clip_by_value(adv_x, -1, 1)
    show_images(adv_x)
```


Different Values of Epsilon

As the value of epsilon is increased, it becomes easier to fool the network. However, this comes as a trade-off which results in the perturbations becoming more identifiable.

```
epsilon: 0
adversarial_cat1.png is cat
  1.000 cat
  0.000 dog
```



```
epsilon: 0.5
adversarial_cat1.png is dog
  0.994 dog
  0.006 cat
```



```
epsilon: 20
adversarial_cat1.png is dog
  0.973 dog
  0.027 cat
```



Final Result

Using FGSM with epsilon = 1.

```
epsilon: 1  
adversarial_cat1.png is dog  
1.000 dog  
0.000 cat
```



```
epsilon: 1  
adversarial_cat2.png is dog  
1.000 dog  
0.000 cat
```



```
epsilon: 1  
adversarial_dog1.png is cat  
1.000 cat  
0.000 dog
```

