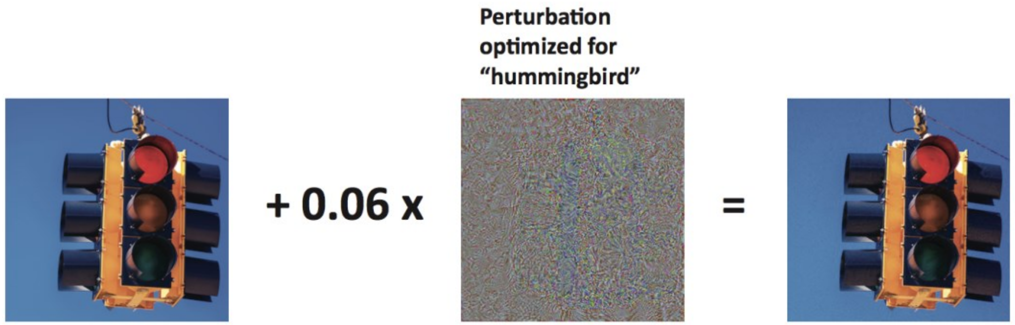
CS5260 Tutorial 1: Adversarial Machine Learning

Xiangyu Peng

Jan 13, 2023

Recap: Adversarial Machine Learning

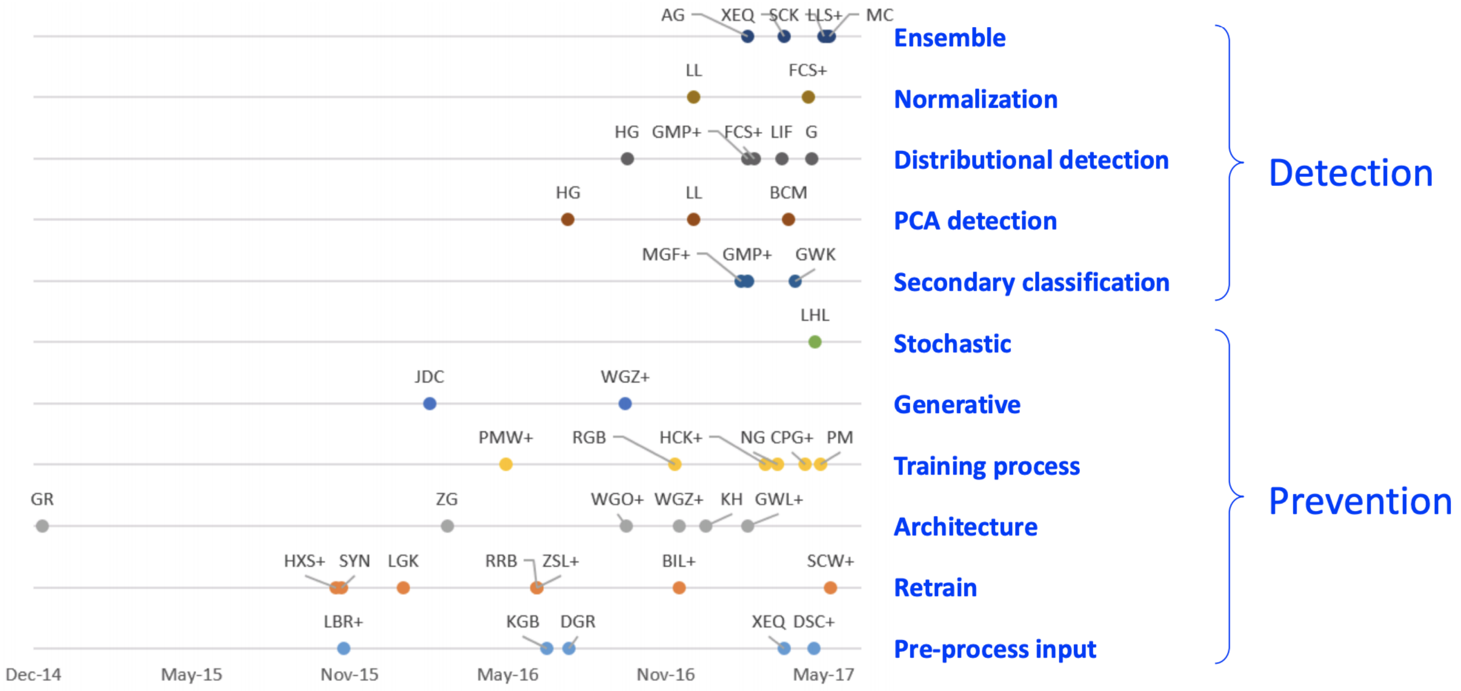
- Explaining and Harnessing Adversarial Examples, ICLR 2015
  - Open a new line of research direction
  - Why do adversarial examples exist?
    - high dimensions
  - How to generate adversarial examples?
    - Fast Gradient Signed Method (**Assignment 1**)
    - White box **VS** black box attacks
  - How to defend adversarial attacks?
    - Adversarial training



The resulting image shown on the right is classified as **"hummingbird"** by a pre-trained Inception V3 network with 99.9% confidence.

You can explore more if interested

Numerous Defenses Proposed



- Slide adapted from KDD 2019 tutorial

Recap: Fast Gradient Signed Method (FMSM)

- When you train NN, training data is fixed while weights of NN are updated using gradient descent

$$w \leftarrow w - \nabla J_w(w, x, y)$$

- Key idea: when you ‘train adversarial examples’, weights are fixed while data is updated using gradient ascent

$$x \leftarrow x + \nabla J_x(w, x, y)$$

Recap: Fast Gradient Signed Method (FGSM)

- Formulation
  - 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.

$$adv\_x = x + \epsilon * \mathrm{sign}(\nabla_x J(\theta, x, y))$$

Assignment 1

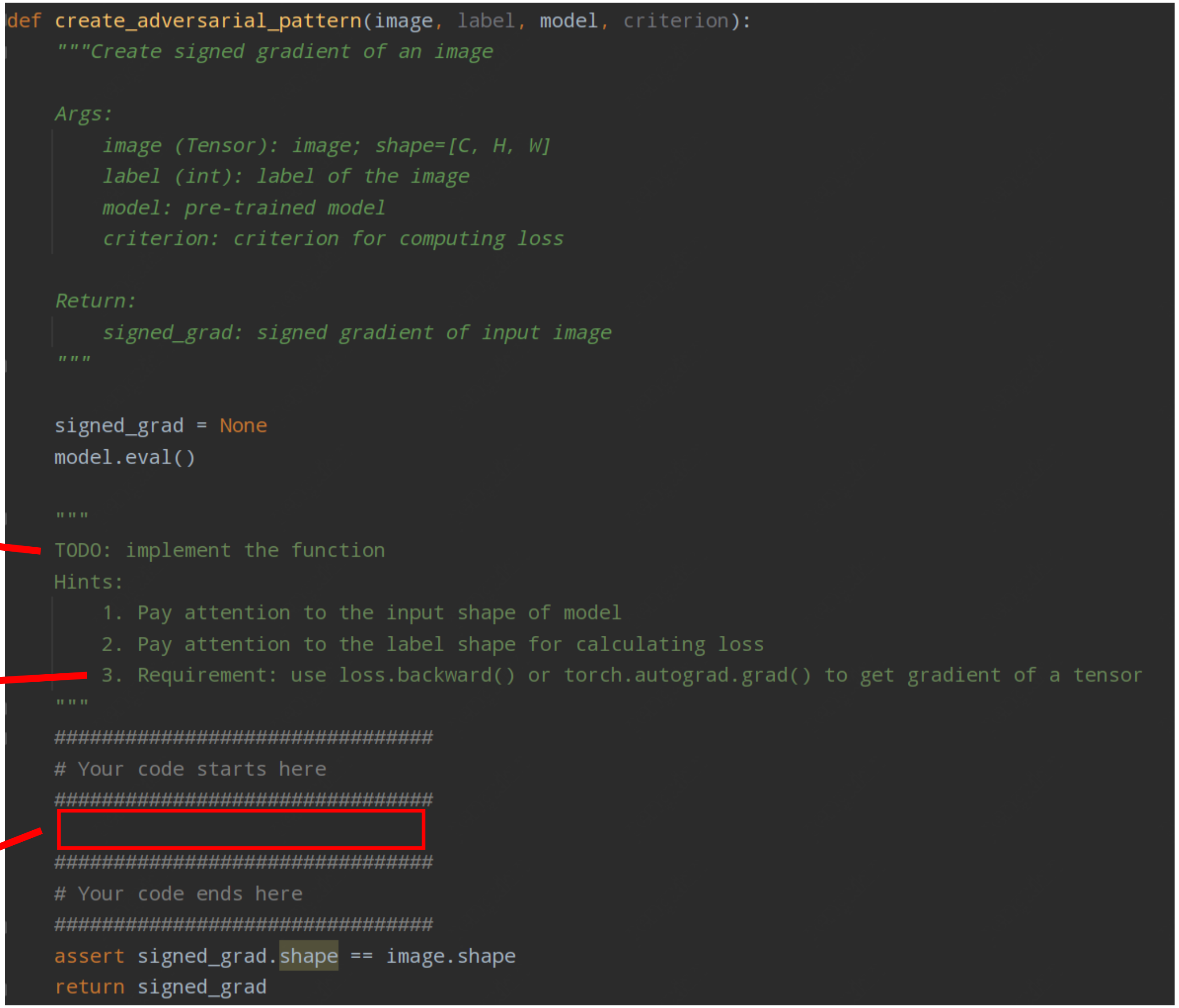
- Implement key functions of FGSM
  - $adv\_x = x + \epsilon * \text{sign}(\nabla_x J(\theta, x, y))$
- The model to attack:
  - ResNet18 pre-trained on ImageNet

Assignment 1

- Example code snippet

Find the key words:

- “TODO” indicates you need to write code below
- “Requirement” means the requirement must be satisfied
- Only write your code in this area, between “Your code starts here” and “Your code ends here”;
- Do not modify anywhere else, including comments



```python
def create_adversarial_pattern(image, label, model, criterion):
    """Create signed gradient of an image

    Args:
        image (Tensor): image; shape=[C, H, W]
        label (int): label of the image
        model: pre-trained model
        criterion: criterion for computing loss

    Return:
        signed_grad: signed gradient of input image
    """

    signed_grad = None
    model.eval()

    """
    TODO: implement the function
    Hints:
        1. Pay attention to the input shape of model
        2. Pay attention to the label shape for calculating loss
        3. Requirement: use loss.backward() or torch.autograd.grad() to get gradient of a tensor
    """
    ################################
    # Your code starts here
    ################################

    ################################
    # Your code ends here
    ################################
    assert signed_grad.shape == image.shape
    return signed_grad
```

Assignment 1

- More to explore if you have time
  - Experiment on different models
  - Use different images
  - Try different attack methods
  - Generalize to other areas, e.g., NLP, Reinforcement Learning
- Still a large open question
- Your own attack or defense methods
  - Could be published in top-tier conference

Q & A