## Lab 33: Using Foolbox for Adversarial Attack

**Using Foolbox for Adversarial Attack**

**Lab Manual**



**Disclaimer: The content is curated from online/offline resources and used for educational purpose only**

***Steps to implement Foolbox Library***

1. Visit the link: <https://colab.google/>
2. Click on ‘New Notebook’
3. Start typing the code given below
4. ***Installing the library***

pip install Foolbox

1. ***Implementing the code for attack and visualization***

# Required installations (run in terminal or notebook if not already done):

# pip install foolbox torch torchvision matplotlib numpy pillow

import torch

import torchvision.models as models

import torchvision.transforms as transforms

from PIL import Image

import foolbox as fb

import numpy as np

import matplotlib.pyplot as plt

# Load pretrained ResNet18 model

model = models.resnet18(pretrained=True).eval()

fmodel = fb.PyTorchModel(model, bounds=(0, 1))

# Load and preprocess image

img\_path = "cat.jpg"  # Make sure this image is in the same folder

img = Image.open(img\_path).resize((224, 224))

transform = transforms.Compose([

    transforms.ToTensor()

])

image = transform(img).unsqueeze(0).clamp(0, 1)

# Predict original label

logits = model(image)

label = torch.argmax(logits, dim=1).item()

# Run FGSM attack

attack = fb.attacks.FGSM()

\_, adv\_image, success = attack(fmodel, image, torch.tensor([label]), epsilons=0.03)

# Predict adversarial label

adv\_logits = model(adv\_image)

adv\_label = torch.argmax(adv\_logits, dim=1).item()

# Load human-readable labels for visualization

import json

import urllib.request

url = "https://raw.githubusercontent.com/pytorch/hub/master/imagenet\_classes.txt"

response = urllib.request.urlopen(url)

categories = [line.strip() for line in response.readlines()]

original\_class = categories[label]

adversarial\_class = categories[adv\_label]

# Convert tensors to numpy for plotting

image\_np = image.squeeze().permute(1, 2, 0).detach().numpy()

adv\_image\_np = adv\_image.squeeze().permute(1, 2, 0).detach().numpy()

# Plot the original and adversarial images side by side

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.imshow(image\_np)

plt.title(f"Original: {original\_class}")

plt.axis('off')

plt.subplot(1, 2, 2)

plt.imshow(adv\_image\_np)

plt.title(f"Adversarial: {adversarial\_class}")

plt.axis('off')

plt.suptitle("Visual Difference: Original vs. Adversarial Image", fontsize=14)

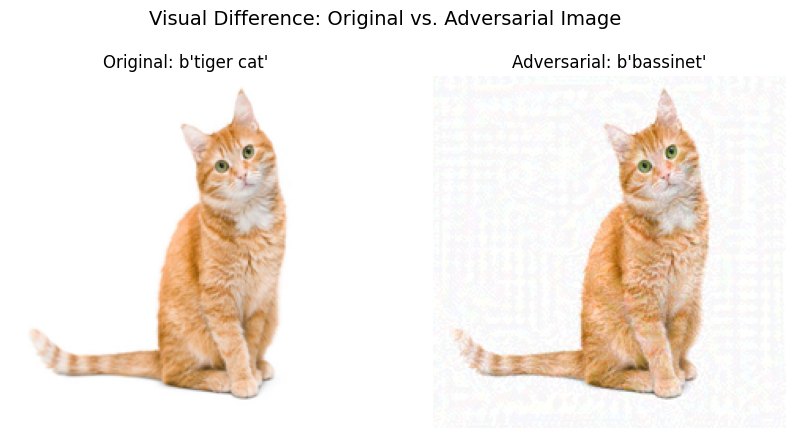
plt.show()

# Print success status

print("Was the attack successful?", success.item())

1. Click on the ***folder icon*** (shown on the left side) and then click on the ***upload icon***.
2. Upload the image of the cat from your device.
3. Now click on ***Run All*** or ***Ctrl + F9*** to run all the cells

**Output:**

****

**Explanation**

**What you see**

* **Left (Original):** This is your real image of a tiger cat. The model correctly predicts the label as tiger cat.
* **Right (Adversarial)**: This image looks the same to a human, but it has tiny invisible changes (perturbations). These are specially crafted to confuse the model. The model now thinks this is a bassinet!
* **Text Below**: Was the attack successful? ***True*** means that the adversarial image successfully fooled the AI into making the wrong prediction.

**What are the changes**

We used the ***FGSM attack (Fast Gradient Sign Method)***.

This method:

* Computes the gradient of the model’s prediction with respect to the input image.
* Adds a small tweak (epsilon=0.03) to the image to change the prediction.

**Try on your own:**

If you want to try, when will the model fail - Make epsilon extremely small so that attack is weak.

Change the epsilon from ***0.03*** to ***1e - 6***

| **Concept** | **Explanation** |
| --- | --- |
| **Adversarial Example** | An image that looks normal but is subtly changed to confuse AI. |
| **Model Vulnerability** | AI models can be easily tricked—even when humans see no difference. |
| **Importance of AI Security** | This is why security and robustness are **critical in real-world AI systems** (e.g., self-driving cars, healthcare). |
| **Epsilon Value** | Controls how strong the attack is. Higher epsilon means more visible change, but easier to fool the model. |