2023 암호분석경진대회

x번 문제

1.Description

1) CIFAR10 데이터 혹은 ImageNet 데이터에 있는 이미지 중에서 적당한 이미지에 대한 PGD(Projected Gradient Descent) 기반의 Perturbation 공격을 수행하라.
성공적인 AI 모델에 대한 공격이 이뤄졌음을 증명하고, 이때, 사람의 육안으로는 공격 전 이미지와 공격 후 이 미지를 (거의) 구분할 수 없음을 보여라.
2) 현실적으로 위협적인 공격 기법으로는 Patch Attack이 존재한다. 교통표지판(STOP 신호, 속도제한 신호 표지판 등)에 대한 Patch Attack 사례를 제시하라. 실제 오인식됨을 증명하라.

2.Solution

1) ImageNet 데이터의 이미지에 대해 PGD(Projected Gradient Descent) 기반의 Perturbation 공격을 수행하고, 성공적인 공격이 이뤄졌음을 증명하기 위해 아래와 같이 실험을 진행하였다.

공격 대상이 되는 객체 분류용 AI 모델로는 pre-trained ResNet-152를 사용했다. 해당 모델은 ImageNet LSVRC 2012 Validation Set에 대해 약 82.1%의 분류 정확도를 보였다.

```
# Test model accuracy
correct,total=0,0
with torch.no_grad():
for images,labels in test_loader:
images=images.to(device)
labels=labels.to(device)
outputs =model(images)
preds =outputs.argmax(1, keepdim =True)
correct +=preds.eq(labels.view_as(preds)).sum()
total +=labels.shape[0]
print(f"model accuracy: {100 *correct /total}")
```

코드 1

```
model accuracy: 82.10000610351562
```

그림 1. 코드 1의 실행 결과

이후 공격을 수행하였다. 공격 대상 이미지는 임의로 선정했다.

아래는 ImageNet 데이터에 있는 hard disc 이미지에 대해 Perturbation 공격을 수행하였더니, 본래의 이미지는 hard disc라고 인식하는 반면, Perturbated 이미지는 스컹크로 인식함을 보여준다. 즉 성공적인 공격이 이루어졌음을 보여준다.

```
# Orinal prediction
imshow(images,categories[labels.item()])
with torch.no_grad():
outputs =model(images)
probabilities =torch.nn.functional.softmax(outputs[0], dim=0)
# Show top categories of original image
top5_prob, top5_catid =torch.topk(probabilities, 5)
x =np.arange(5)
top5_categories=[]
top5_probabilities=[]
for i in range(top5_prob.size(0)):
top5_categories.append(categories[top5_catid[i]])
top5_probabilities.append(top5_prob[i].item())
plt.bar(x, top5_probabilities)
plt.xticks(x, top5_categories)
plt.show()
```

코드 2

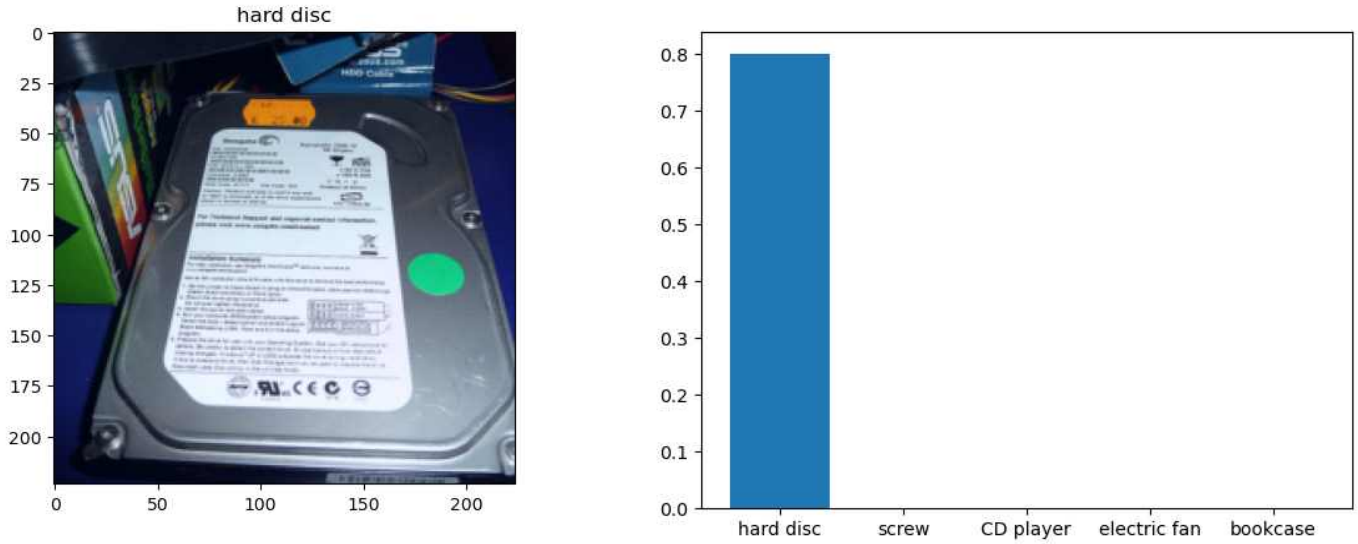

그림 2. 코드 2의 실행 결과: hard disc 사진(위 사진)이 ResNet-152을 통과하면 해당 모델은 높은 신뢰도로 클래스 'hard disc'로 인식한다(위 그래프).

```
# Perform an attack
atk = PGD(model, eps=8/255, alpha=2/225, steps=10, random_start=True)
atk.set_normalization_used(mean=MEAN, std=STD)
print('Perturbation Attack')
print(atk)
perturbated_images =atk(images, labels)
# Perturbated prediction
imshow(perturbated_images,'perturbated '+categories[labels.item()])
with torch.no_grad():
outputs =model(perturbated_images)
probabilities =torch.nn.functional.softmax(outputs[0], dim=0)
# Show top categories of perturbated image
top5_prob, top5_catid =torch.topk(probabilities, 5)
x =np.arange(5)
top5_categories=[]
top5_probabilities=[]
for i in range(top5_prob.size(0)):
top5_categories.append(categories[top5_catid[i]])
top5_probabilities.append(top5_prob[i].item())
plt.bar(x, top5_probabilities)
plt.xticks(x, top5_categories)
plt.show()
```

코드 3

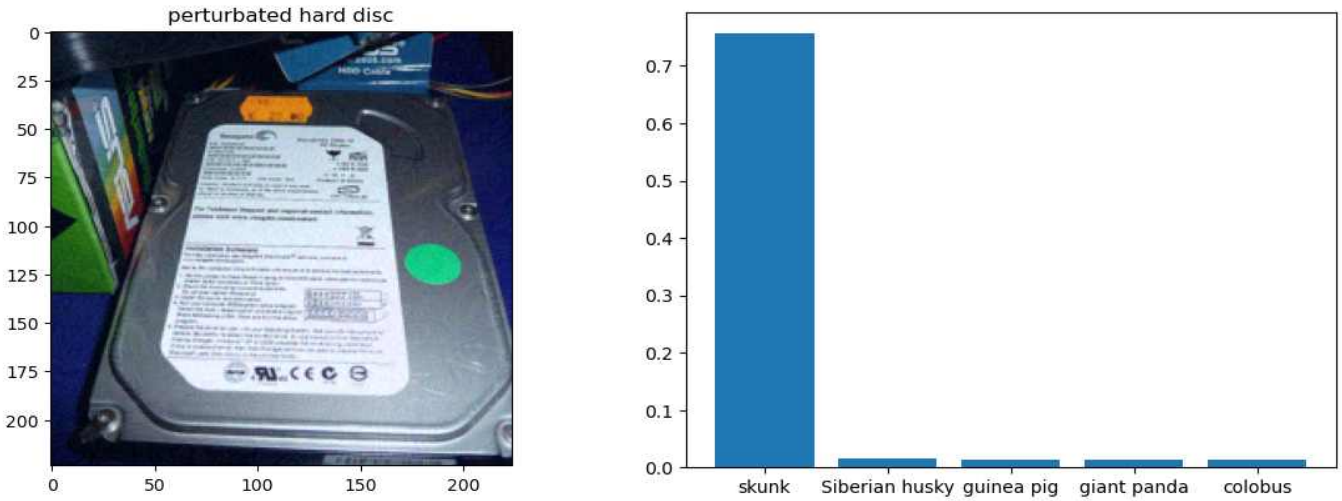

그림 3. 코드 3의 실행 결과 :perturbated hard disc 사진(위 사진)이 ResNet-152을 통과하면 해당 모델은 높은 신뢰도로 클래스 'skunk'로 인식한다(위 그래프).

본래의 이미지와 Perturbated 이미지는 육안으로 거의 구분될 수 없음을 그림 4를 통해 확인 할 수 있다.

그림 4. 원본 이미지(좌)와 perturbated 이미지(우)

2) 교통표지판에 대해 Patch Attack을 수행하고, 성공적인 공격이 이뤄졌음을 증명하기 위해 아래와 같이 실험을 진행하였다.

분류자를 훈련시키고 공격을 수행하기에 앞서, 해당 문제에서 사용할 교통표지판 데이터는 GTSRB - German Traffic Sign Recognition Benchmark(독일교통표지판)로 선정하였다. 해당 데이터 셋은 40개 이상의 클래스로 분류된 총 50,000개 이상의 이미지로 이루어져 있으며, 국제 신경망 공동회의(IJCNN) 2011에서 개최된 다중 클래스 단일 이미지 분류 과제로 사용되기도 하였다.

그림 5. GTSRB
이미지 예시

먼저 GTSRB(독일교통표지판) 데이터 분류를 위한 AI을 만들었다. 전이학습을 이용하여 훈련하였으며, 모델을 pre-trained ResNet-18으로 세팅한 후 finetuning을 진행하였다. 해당 모델은 약 99.9%의 분류 정확도를 보였다.

```
# %%
# val model accuracy
model.load_state_dict(torch.load(PATH_TO_MODEL))
correct,total=0,0
with torch.no_grad():
for images,labels in val_loader:
images=images.to(device)
labels=labels.to(device)
outputs =model(images)
preds =outputs.argmax(1, keepdim =True)
correct +=preds.eq(labels.view_as(preds)).sum()
total +=labels.shape[0]
print(f"model accuracy: {100 *correct /total}")
```

코드 4

```
model accuracy: 99.93624114990234
```

그림 6. 코드 4의 실행 결과

이후 공격을 수행하였다.

앞서 훈련한 분류자 모델과 GTSRB 데이터를 사용하여 Adversarial Patch를 생성하였다. Adversarial Patch의 타겟, 즉 해당 패치를 붙였을 때 분류자 모델이 인식해야할 결과는 Go Straight Sign(ClassID=35)으로, Adversarial Patch의 넓이는 이미지의 20%로 설정하였다.

```
LR =1.0
EPOCHS =5
THRESHOLD=0.9
MAX_ITER=100
TARGET=35 # Go Straight Traffic Sign
# Initialize the patch
IMG_SIZE=(3, 112, 112)
NOISE_RATIO=0.2
MASK_SIZE =int((NOISE_RATIO *IMG_SIZE[1] *IMG_SIZE[2])**0.5)
patch =np.random.rand(IMG_SIZE[0], MASK_SIZE, MASK_SIZE)
```

코드 5

Adversarial Patch를 얻기 위해, 패치를 초기화한 후 다음의 함수를 최대화하도록 학습시켰다.

$$\mathbb{E}x \sim X, l \sim L[log(Pr(TARGET|A(patch, x, l)))]$$

여기서 A(p, x, l)는 이미지 x의 위치 x에 패치 p를 붙이는 연산자이며, Pr(y|x)는 입력 이미지 x에 대해 분류자 모델이 y로 인식할 확률이고, X는 이미지의 훈련 셋, L은 이미지의 위치에 대한 분포를 나타낸다.

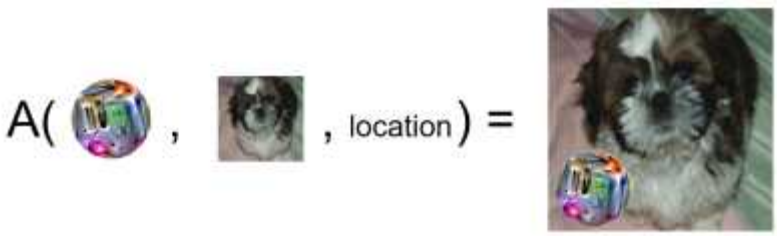

그림 7. A(p, x, l) 연산의 예시

이를 풀어 설명하자면, 이미지를 뽑으면서 위치를 바꿔가면서 패치를 적용해보아도 이 패치가 여전히 그 모든 전체 결과에 대해서 Adversarial Patch로써 잘 동작할 수 있도록 학습을 시키는 것이다.

해당 패치는 그림 9와 같이 약 99.0%의 공격 성공률을 보였다.

```
imshow(best_patch,"Adversarial Patch")
print("Success Rate = %.4f"%(best_acc))
```

코드 6

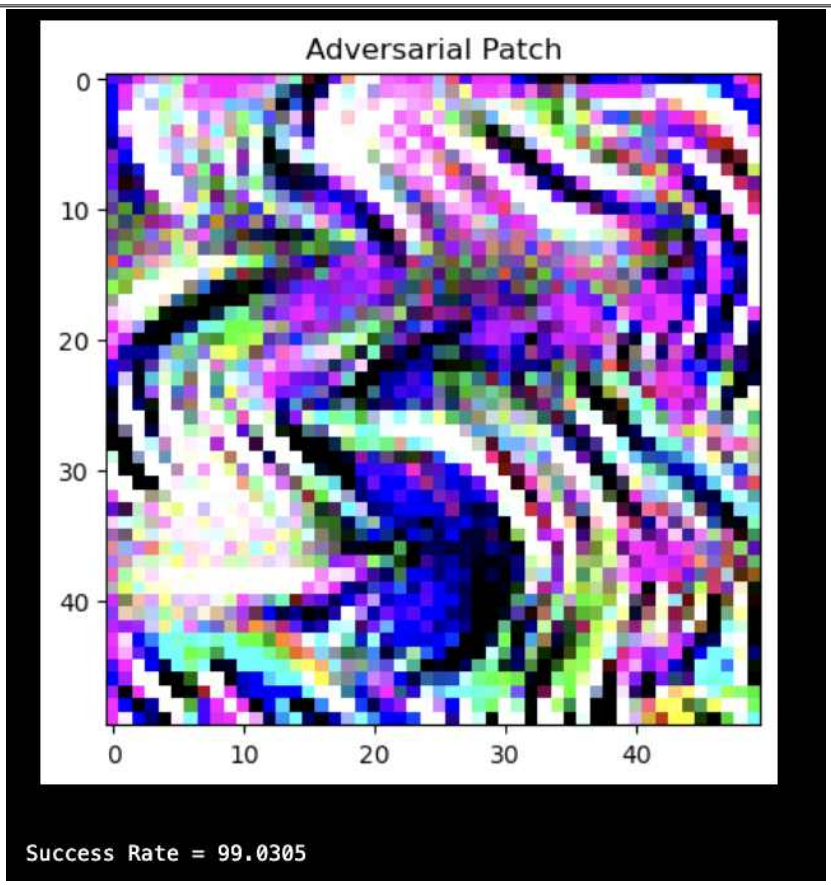

그림 8. 코드 6의 실행 결과

아래는 GTSRB 데이터에 있는 20 Speed Limit Sign(ClassID=0) 이미지에 위에서 만든 Adversarial Patch를 붙였더니 본래의 이미지는 20 Speed Limit Sign(ClassID=0)이라고 인식하는 반면, 패치를 붙인 이미지는 Go Straight Sign(ClassID=35)으로 오인식함을 보여준다. 즉 성공적으로 공격이 수행되었음을 보여준다.

```
# Orinal prediction
imshow(images.squeeze(0),str(labels.item()))
with torch.no_grad():
outputs =model(images)
probabilities =torch.nn.functional.softmax(outputs[0], dim=0)
# Show top categories of original image
top5_prob, top5_catid =torch.topk(probabilities, 5)
x =np.arange(5)
top5_categories=[]
top5_probabilities=[]
for i in range(top5_prob.size(0)):
top5_categories.append(str(top5_catid[i].item()))
top5_probabilities.append(top5_prob[i].item())
plt.bar(x, top5_probabilities)
plt.xticks(x, top5_categories)
plt.show()
```

코드 7

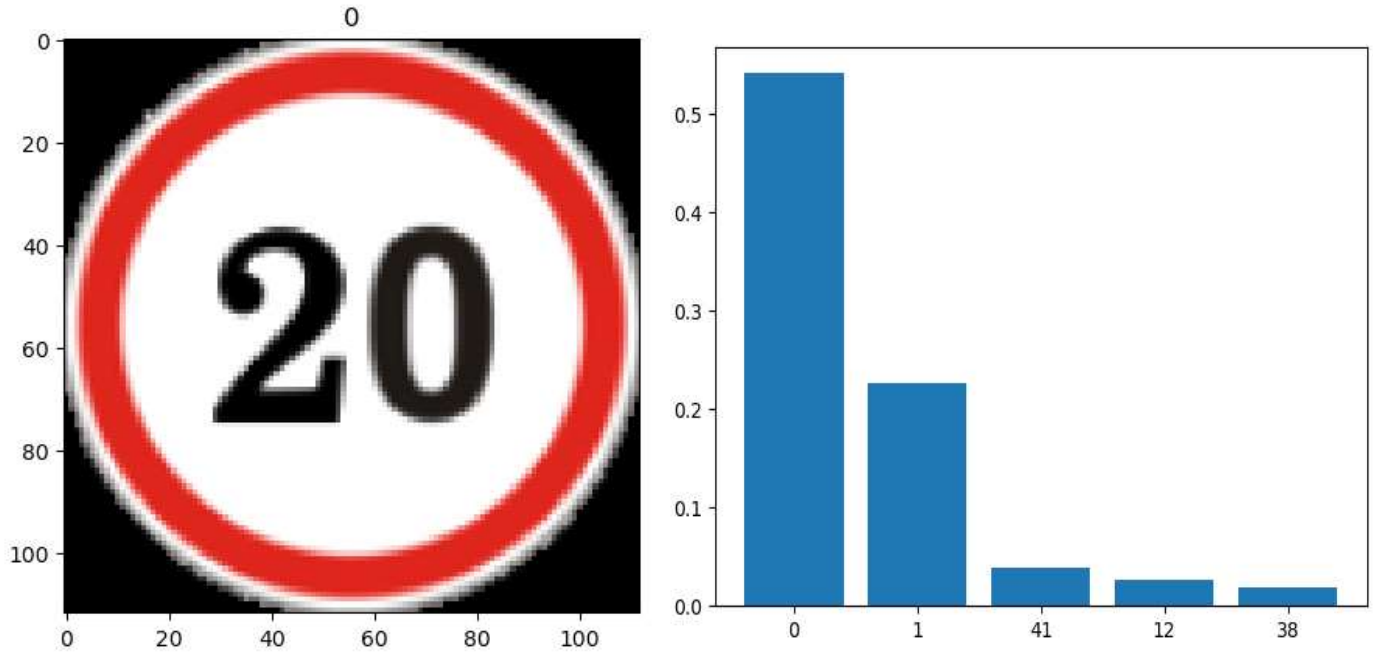

그림 9. 코드 7의 실행 결과: 20 Speed Limit Sign(ClassID=0) 이미지(위 사진)가 분류자 모델을 통과하면, 해당 모델은 높은 신뢰도로 ClassID가 '0'이라고 분류한다(위 그래프).

```
# Apply the patch
extended_patch =np.zeros(IMG_SIZE)
x_loc, y_loc =np.random.randint(low=0, high=IMG_SIZE[1]-best_patch.shape[1]),
np.random.randint(low=0, high=IMG_SIZE[2]-patch.shape[2]) # random patch location
for i in range(best_patch.shape[0]):
extended_patch[:, x_loc:x_loc +best_patch.shape[1], y_loc:y_loc +best_patch.shape[2]] =best_patch
mask =extended_patch.copy()
mask[mask !=0] =1.0
extended_patch =torch.from_numpy(extended_patch)
mask =torch.from_numpy(mask)
patched_images =torch.mul(mask.type(torch.FloatTensor), extended_patch.type(torch.FloatTensor))
+torch.mul((1 -mask.type(torch.FloatTensor)), images.type(torch.FloatTensor))
# Patched prediction
imshow(patched_images.squeeze(0),'patched '+str(labels.item()))
with torch.no_grad():
outputs =model(patched_images)
probabilities =torch.nn.functional.softmax(outputs[0], dim=0)
# Show top categories of perturbated image
top5_prob, top5_catid =torch.topk(probabilities, 5)
x =np.arange(5)
top5_categories=[]
top5_probabilities=[]
for i in range(top5_prob.size(0)):
top5_categories.append(str(top5_catid[i].item()))
top5_probabilities.append(top5_prob[i].item())
plt.bar(x, top5_probabilities)
plt.xticks(x, top5_categories)
plt.show()
```

코드 8

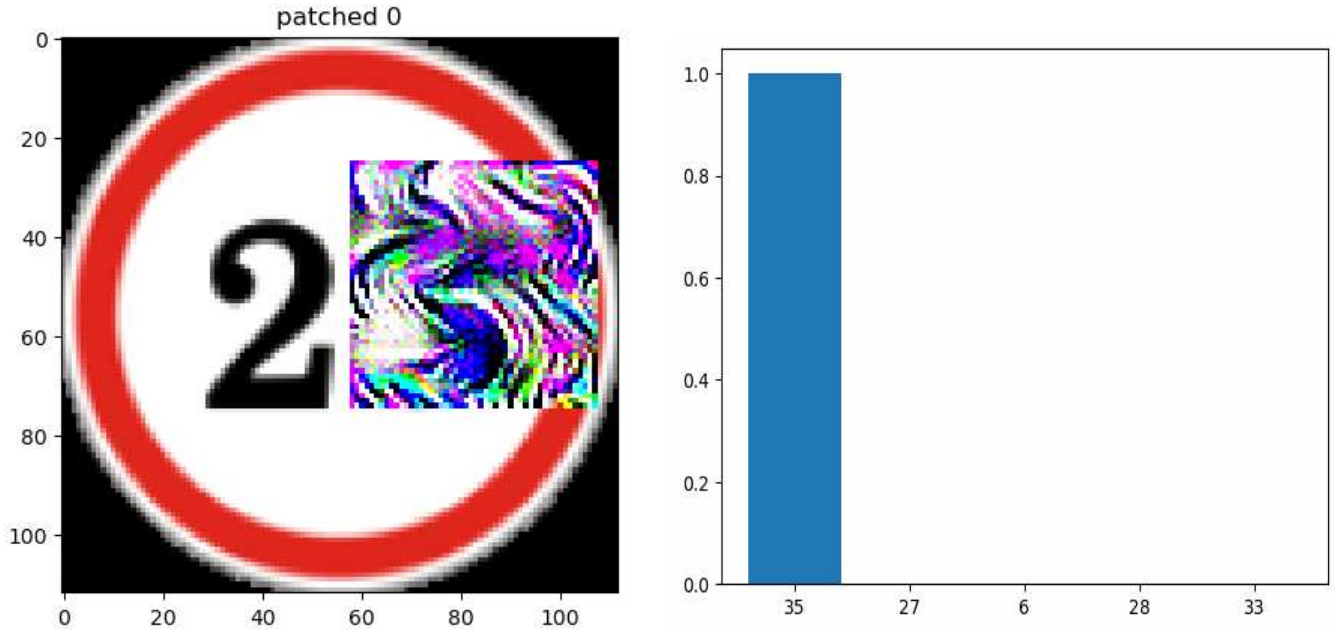

그림 10. 코드 8의 실행 결과: : Go Straight Sign(ClassID=35)을 대상으로 하는 Adversarial Patch를 그림 10과 동일한 20 Speed Limit Sign(ClassID=0) 이미지에 부착하면 해당 이미지(위 사진) 은 높은 신뢰도로 ClassID가 '35'로 분류된다(위 그래프).

본 답안을 작성하는 데 사용된 전체코드 및 그 실행 결과는 첨부된 파일에서 확인할 수 있다.