

캡스톤 디자인 ‘딥페이크 탐지’

#14. Perturbation(GoogleNet) & Unsharp Mask

김지수, 김민지, 민지민

지난주







- PyTorch의 perturbation Official Code는 없었음
- 5~6년 전에 개발된 TensorFlow의 perturbation Official Code 찾음
- 버전 문제가 심했고, 코드가 상세하여 이해하기 어려웠고, 오류 해결이 끝나지 않았음
- 계산된 perturbation을 사용하는 게 더 낫다고 생각하여
→ 현재 사용 중인 xception과 비슷한 googlenet 계열을 통해 생성된 perturbation을 찾아서 적용하려고 함

Perturbation

Generalizable Data-free Objective for Crafting Universal Adversarial Perturbations

Konda Reddy Mopuri*, Aditya Ganeshan*, R. Venkatesh Babu, *Senior Member, IEEE*

- <https://arxiv.org/pdf/1801.08092.pdf> -> 저자의 코드 구현 <https://github.com/val-iisc/GD-UAP>
- github에 precomputed perturbation.npy 파일 존재 →
- googlenet_no_data.npy 선택하여 사용

	googlenet_no_data.npy
	googlenet_with_data.npy
	googlenet_with_range.npy
	resnet152_no_data.npy
	resnet152_with_data.npy
	resnet152_with_range.npy

Perturbation

```
class UniversalPerturbation(Function):  
    def forward(ctx, image):  
        pert_path = 'E:/df_dataset/perturbations/universal.npy'  
        pert_ndarr = np.load(pert_path)  
        pert_ndarr = np.squeeze(pert_ndarr)  
  
        pert_img = Image.fromarray(pert_ndarr, 'RGB')  
        pert_img = pert_img.resize((960, 540))  
        np_pert = np.asarray(pert_img)  
  
        img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  
        np_human = np.asarray(img)  
  
        (start, end) = get_attack_params('universal_perturbation')['level']  
        level = np.random.uniform(start, end)  
  
        np_pert = np_pert * level  
        np_pert = np_pert.astype(int)  
        np_pert = np.clip(np_pert, 0, 255)  
  
        np_plus = np_pert + np_human.astype(int)  
        np_plus = np.clip(np_plus, 0, 255)  
  
        plus_img = Image.fromarray(np_plus.astype('uint8'), 'RGB')  
        return plus_img
```

- numpy array인 perturbation을 이미지로 변환해 resize 후 다시 numpy array로 변환
- perturbation array에 0.05~0.5 사이의 강도 값을 곱한 후 원래 이미지와 더함

Perturbation - 생성한 이미지



Lv1(0.05)



Lv2(0.1)

Perturbation - 생성한 이미지

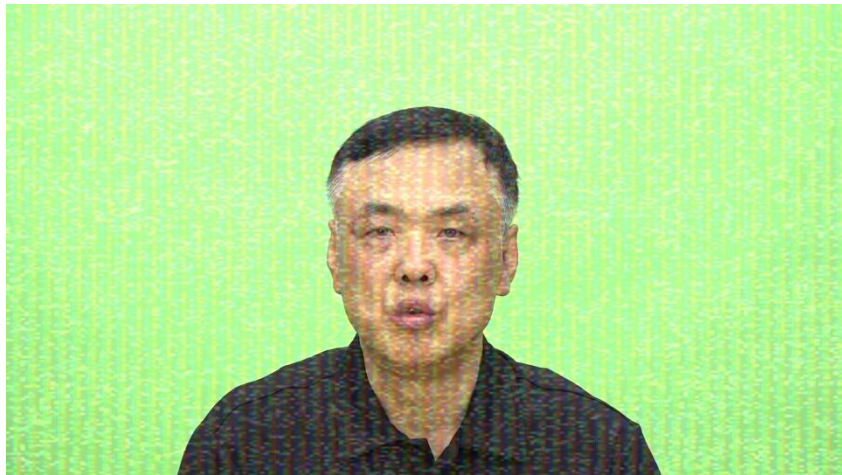


Lv3(0.15)

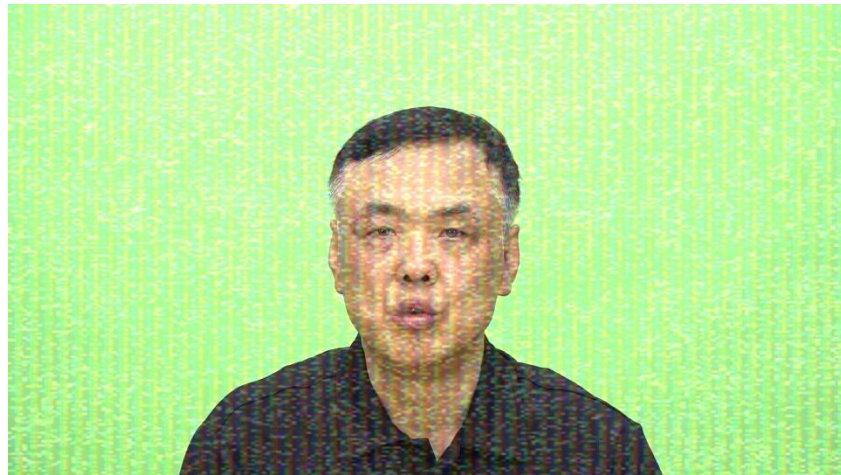


Lv4(0.2)

Perturbation - 생성한 이미지



Lv5(0.25)



Lv6(0.3)

Perturbation - 생성한 이미지



Lv7(0.35)



Lv8(0.4)

Perturbation - 생성한 이미지



Lv9(0.45)



Lv10(0.5)

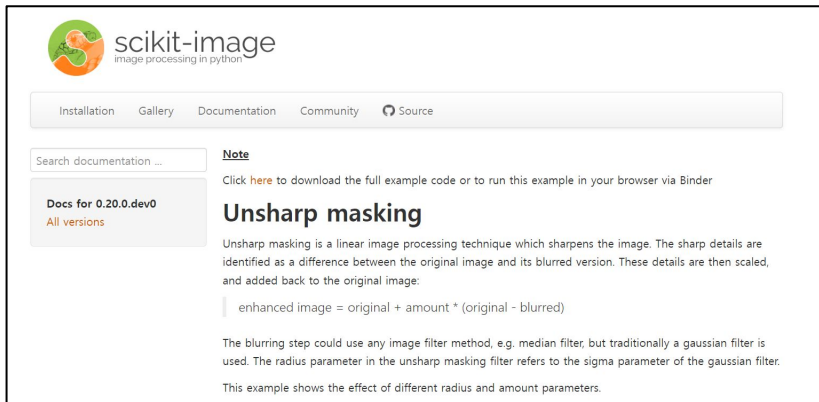
PSNR: 29.2

Perturbation - inference(원본 데이터셋만 학습한 모델)

level1	loss- 0.0049, acc- 0.998
level2	loss- 0.0228, acc- 0.995
level3	loss- 0.0729, acc- 0.991
level4	loss- 0.1727, acc- 0.985
level5	loss- 0.3194, acc- 0.964
level6	loss- 0.4787, acc- 0.869
level7	loss- 0.5839, acc- 0.732
level8	loss- 0.6374, acc- 0.639
level9	loss- 0.6573, acc- 0.602
level10	loss- 0.6662, acc- 0.578

Sharpening(Unsharp Mask 사용)

- 이전 방식에 해결하기 힘든 오류가 발생하여
- scikit-image 패키지의 unsharp_mask 내장함수 사용



The screenshot shows the scikit-image documentation page for 'Unsharp masking'. The page header includes the scikit-image logo and navigation links: Installation, Gallery, Documentation, Community, and Source. A search bar is present. The main content area is titled 'Unsharp masking' and includes a 'Note' section with a link to download example code or run it via Binder. The text explains that unsharp masking is a linear image processing technique that sharpens an image by identifying differences between the original and a blurred version, then scaling and adding them back. A formula is provided:
$$\text{enhanced image} = \text{original} + \text{amount} * (\text{original} - \text{blurred})$$
 The text also mentions that the blurring step could use any image filter method, such as a median filter, but traditionally a gaussian filter is used, and that the radius parameter refers to the sigma parameter of the gaussian filter. Finally, it states that the example shows the effect of different radius and amount parameters.

```
from skimage import data
from skimage.filters import unsharp_mask
import matplotlib.pyplot as plt

image = data.moon()
result_1 = unsharp_mask(image, radius=1, amount=1)
result_2 = unsharp_mask(image, radius=5, amount=2)
result_3 = unsharp_mask(image, radius=20, amount=1)
```

Sharpening(Unsharp Mask 사용) - 원본 데이터셋만 학습한 모델로 추론

level1	loss- 0.0023, acc- 1.000
level2	loss- 0.2095, acc- 0.938
level3	loss- 0.4947, acc- 0.901
level4	loss- 0.7934, acc- 0.856
level5	loss- 1.1414, acc- 0.789
level6	loss- 1.5709, acc- 0.701
level7	loss- 2.0184, acc- 0.631
level8	loss- 2.4498, acc- 0.569
level9	loss- 2.8313, acc- 0.528
level10	loss- 3.1511, acc- 0.511

random combine attack 공격 모델 생성

이전에 언급한 사항 제외하고는 전 주와 동일하게 실행

```
-----  
  
TypeError                                Traceback (most recent call last)  
  
<ipython-input-98-b475f2e795dc> in <module>  
----> 1 go(real_train_list, fake_train_list, real_valid_list, fake_valid_list)  
      2  
  
<ipython-input-97-a9fab4d1893a> in go(real_train_list, fake_train_list, real_valid_list, fake_valid_list)  
    12     train_realnoise = cv2.imread(img)  
    13     for attack in attacks:  
--> 14         train_realnoise = attack(train_realnoise)  
    15     # JPEG 적용  
    16     (start, end) = get_attack_params('jpeg')['quality']  
  
TypeError: FunctionBackward.forward: expected Tensor or tuple of Tensor (got numpy.ndarray) for return value 0
```

질문 사항

jpeg 마지막에 적용하는 이유

PSNR 29 정도 나오는데 사용해도 되는지

후속연구로 진행하는건데, 직전 연구에 대한 설명을 어느 정도 넣어야하는지

(단일 공격 학습 모델의 경우) -> 학습 방법에 직전 연구 내용을 추가해도 되는지?

소개에도 추가해도 되는지?

test시에 한 이미지에 모든 공격들 넣어서 추론하는 방법