캡스톤 디자인
‘딥페이크 탐지’

#7. 블랙박스 공격 조사 및 데이콘 코드 구현

김지수, 김민지, 민지민

지난 캡스톤 회의 내용 -딥페이크 탐지기 성능 및 안티포렌식 기법 조사-

FGSM : 엡실론 값이 클수록 노이즈가 눈에 띄어 잘못된 예측 가능

PGD : FGSM 응용한 방법으로, step 수에 따라 공격 강도 강해짐

Deepfake Detector : train 5000장, test 2000장으로 성능측정

(0.98~0.99 accuracy)

sharpening, denoising, jpeg compression 변형 방법으로 성능 측정했지만, 성능에 영향을 끼치지 않았었음.

데이터셋 가져오기

- 지난 번과 동일한 방식으로 인물 별 생성방식(3) 당 하위폴더 200개 이상
  → 하위 폴더 70개에서 이미지 1장씩 선택

- Train – 5250장
- Valid – 2100장

denoise

◆ fastNlMeansDenoisingColored() [2/2]

```
void cv::fastNlMeansDenoisingColored ( InputArray  src,
                                       OutputArray dst,
                                       float       h = 3 ,
                                       float       hColor = 3 ,
                                       int         templateWindowSize = 7 ,
                                       int         searchWindowSize = 21
                                     )
```

Python:

cv.fastNlMeansDenoisingColored(src[, dst[, h[, hColor[, templateWindowSize[, searchWindowSize]]]]]) -> dst

`#include <opencv2/photo.hpp>`

Modification of fastNlMeansDenoising function for colored images.

Parameters

src	Input 8-bit 3-channel image.
dst	Output image with the same size and type as src .
templateWindowSize	Size in pixels of the template patch that is used to compute weights. Should be odd. Recommended value 7 pixels
searchWindowSize	Size in pixels of the window that is used to compute weighted average for given pixel. Should be odd. Affect performance linearly: greater searchWindowsSize - greater denoising time. Recommended value 21 pixels
h	Parameter regulating filter strength for luminance component. Bigger h value perfectly removes noise but also removes image details, smaller h value preserves details but also preserves some noise
hColor	The same as h but for color components. For most images value equals 10 will be enough to remove colored noise and do not distort colors

The function converts image to CIELAB colorspace and then separately denoise L and AB components with given h parameters using fastNlMeansDenoising function.

- h: 필터의 강도를 결정하는 인자
- hColor: h와 동일한데, 컬리이미지에서 작동
- templateWindowSize: 홀수이어야 함
- searchWindowSize: 홀수이어야 함

denoise

강도: 3

강도: 5

강도: 10

sharpening

```
sharpening_arr = np.array([[-1, -1, -1, -1, -1],
                           [-1, 2, 2, 2, -1],
                           [-1, 2, 9, 2, -1],
                           [-1, 2, 2, 2, -1],
                           [-1, -1, -1, -1, -1]]) / 9.0
```

중앙값이 클수록 강한 샤프닝 적용

```
dst = cv2.filter2D(image_RGB, -1, sharpening_arr)
```

sharpening

중앙값: 9

```
strong_sharpening_arr = np.array([[-1, -1, -1],
                                  [-1, 9, -1],
                                  [-1, -1, -1]])
```

중앙값: 7

```
medium_sharpening_arr = np.array([[0, -1.5, 0],
                                  [-1.5, 7, -1.5],
                                  [0, -1.5, 0]])
```

중앙값: 5

```
weak_sharpening_arr = np.array([[0, -1, 0],
                                [-1, 5, -1],
                                [0, -1, 0]])
```

Jpeg compression

```
image.save(train_dst_path + '/' +  i[:-4] + '_20.jpg', "JPEG", quality = 20)
```

Quality: 50

Quality: 30

Quality: 10

gaussian noise

```python
def gauss_noise(image, gauss_var=1000):
    mean = 0
    sigma = gauss_var ** 0.5
    gauss = np.random.normal(mean, sigma, image.shape)

    res = image + gauss
    noisy = np.clip(res, 0, 255).astype(np.uint8)
    return noisy
```

gaussian noise 함수

gauss_var : 50

gaussian noise

gauss_var :

gauss_var :

Salt-and-pepper Noise

```
def salt_and_pepper(image,p):
    output = np.zeros(image.shape, np.uint8)
    thres = 1-p
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            rdn = random()
            if rdn < p:
                output[i][j] = 0
            elif rdn > thres:
                output[i][j] = 255
            else:
                output[i][j] = image[i][j]
    return output
```

Salt-and-pepper noise 추가 함수

p = 0.001

Salt-and-pepper Noise

p = 0.003

p = 0.005

Deepfake Detector - Dacon 2nd 코드

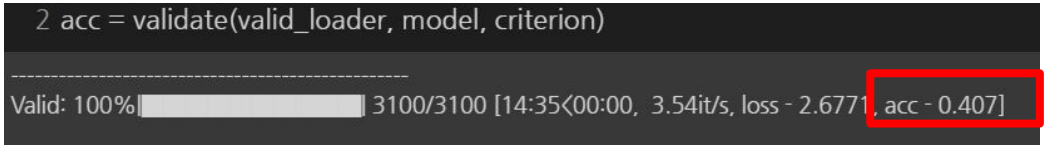
<deepfake 원본 학습코드>

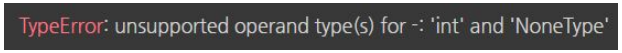
```
-----------------------------------------------
Epoch 1/3
Train:   0%|          | 0/227 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloa
  cpuset_checked))
Train: 100%|          | 227/227 [08:50<00:00,  2.34s/it, loss - 0.0267, acc - 0.992]
Valid: 100%|          | 97/97 [01:31<00:00,  1.06it/s, loss - 0.0806, acc - 0.967]
Epoch 2/3
Train: 100%|          | 227/227 [07:52<00:00,  2.08s/it, loss - 0.0001, acc - 1.000]
Valid: 100%|          | 97/97 [01:13<00:00,  1.33it/s, loss - 0.0600, acc - 0.979]
Epoch 3/3
Train: 100%|          | 227/227 [07:54<00:00,  2.09s/it, loss - 0.0001, acc - 1.000]
Valid: 100%|          | 97/97 [01:13<00:00,  1.33it/s, loss - 0.0395, acc - 0.988]
```

- **epoch 3**으로 학습
- 학습한 모델을 불러와 inference 코드로 안티포렌식 적용 이미지에 대한 결과 확인

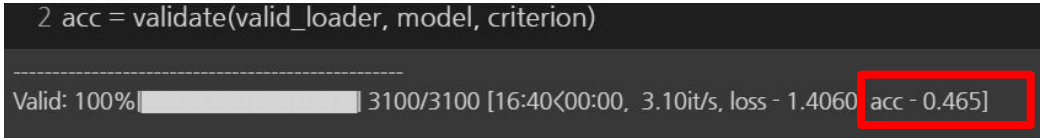
Deepfake Detector - sharpening

weak

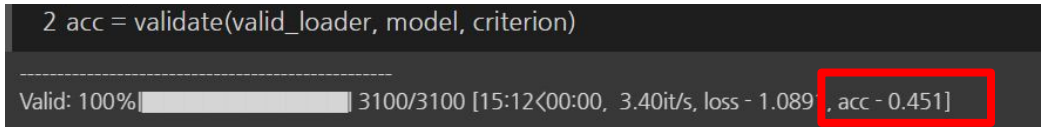




medium



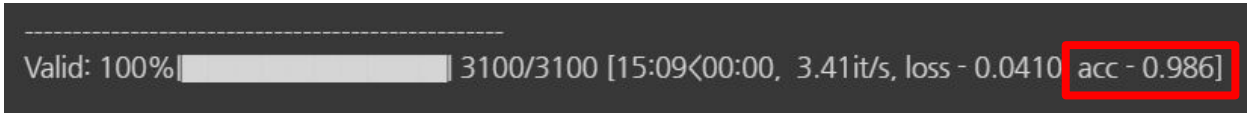
strong



- deepfake 원본에 비해 **성능 하락을 확인**
- 강도별 차이가 크지는 않지만, 강도(약)이 가장 탐지성능이 낮음을 확인

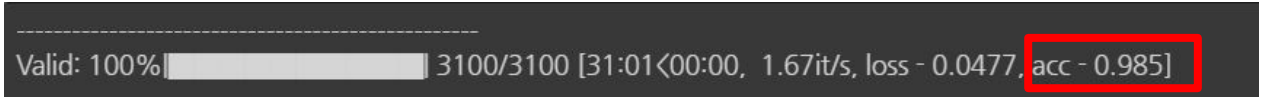
Deepfake Detector - jpeg compression

weak



medium

strong



- deepfake 원본에 비해 **성능 하락하지 않음을 확인**

Deepfake Detector - Salt-and-pepper noise

weak

```
1 acc = validate(valid_loader, model, criterion)
Valid: 100%| 3100/3100 [12:45<00:00,  4.05it/s, loss - 4.682, acc - 0.312]
```

PSNR : 48.060629398643556

medium

```
1 acc = validate(valid_loader, model, criterion)
Valid: 100%| 3100/3100 [12:54<00:00,  4.00it/s, loss - 1.2334 acc - 0.375]
```

PSNR : 43.75484736198632

strong

```
1 acc = validate(valid_loader, model, criterion)
Valid: 100%| 3100/3100 [34:48<00:00,  1.48it/s, loss - 0.4670, acc - 0.574]
```

PSNR : 41.67160011391672

- deepfake 원본에 비해 salt-and-pepper noise 추가 후 **성능 하락을 확인**
- 강도(약)이 가장 탐지성능이 낮음을 확인

Deepfake Detector - gaussian noise

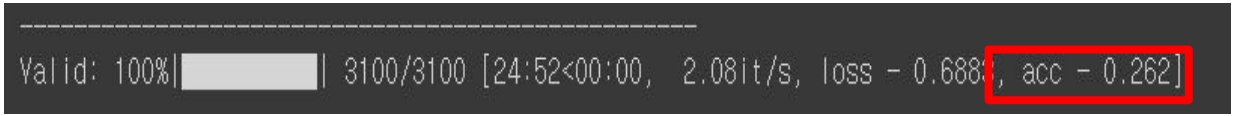
weak

PSNR : 34.05640122656471

medium

PSNR : 31.93704130224397

strong



PSNR : 28.833477927495768

- deepfake 원본에 비해 **성능 하락을 확인**
- acc가 모두 동일하게 나옴 → ?

Deepfake Detector - Dacon 1st 코드

여러 문제로 실행하지 못함. https://github.com/since2020ape/dacon_deepfake

1. **tensorflow 설치 오류**
   - no module named ‘ tensorflow.python’ 오류
   - 여러 방식으로 install 시도해봤지만 계속 동일한 오류 발생


```
C:\Users\user\PycharmProjects\deepFakeAntiForensic
(venv) λ piython3 dacon_deepfake/crop_face_dacon.py E:/deepfake_1st E:/crop_face
Traceback (most recent call last):
  File "dacon_deepfake/crop_face_dacon.py", line 1, in <module>
    import tensorflow as tf
  File "C:\Users\user\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.8_qbz5n2kfra8p0\LocalCache\local-packages\Python38\site-packages\tensorf
    from tensorflow.python.tools import module_util as _module_util
ModuleNotFoundError: No module named 'tensorflow.python'
```