캡스톤 디자인

‘딥페이크 탐지’

#13. 다른 공격 기법 조사 및 inference

김지수, 김민지, 민지민

지난 캡스톤 회의 내용

더 많은 공격(noise)에 대해 조사

-> 학습에 반영하지 않았던 공격에 대한 강인성 확인

- 노이즈 : 포아송 노이즈, 임펄스 노이즈
- 블러 : 미디언 블러, 가우시안 블러, 바이레터럴 필터
- 샤프닝 : 다른 기법

poisson noise

```python
def poisson_noise(image, factor):
    factor = 1 / factor
    img = np.array(image)
    img = img.astype('int16')
    img_noise = np.random.poisson(img * factor) / float(factor)
    np.clip(img_noise, 0, 255, img_noise)
    img_noise = img_noise.astype('uint8')
    return img_noise
```

factor로 강도 조정

poisson level

level 1	0.2
level 2	0.4
level 3	0.6
level 4	0.8
level 5	1
level 6	1.2
level 7	1.4
level 8	1.6
level 9	1.8
level 10	2

poisson level별 이미지

level 1

level 2

poisson level별 이미지

level 3

level 4

poisson level별 이미지

level 5

level 6

poisson level별 이미지

level 7

level 8

poisson level별 이미지

level 9

level 10

poisson inference

	loss	acc
level 1 (0.2)	0.0673	0.983
level 2 (0.4)	0.0563	0.987
level 3 (0.6)	0.0730	0.980
level 4 (0.8)	0.0974	0.965
level 5 (1)	0.1335	0.945
level 6 (1.2)	0.1707	0.930
level 7 (1.4)	0.2386	0.904
level 8 (1.6)	0.2715	0.891
level 9 (1.8)	0.3241	0.857
level 10 (2)	0.3729	0.848

impulse noise

```
def addRvinGray(image,n): # add random valued impulse noise in grayscale
    '''parameters:
        image: type=numpy array. input image in which you want add noise.
        n:  noise level (in percentage)'''
    k=0                  # counter variable
    ih=image.shape[0]
    iw=image.shape[1]
    noisypixels=(ih*iw*n)/100      # here we calculate the number of pixels to

    for i in range(ih*iw):
        if k<noisypixels:
                image[r.randrange(0,ih)][r.randrange(0,iw)]=r.randrange(0,256)
                k+=1
        else:
            break
    return image
```

- addRvinGray 함수를 이용하여 impulse noise 적용
- n 값을 이용해 강도 조절

impulse noise level

level 1	1
level 2	3
level 3	5
level 4	6
level 5	7
level 6	9
level 7	11
level 8	13
level 9	15
level 10	17

impulse noise level별 이미지

level 1

level 2

impulse noise level별 이미지

level 3

level 4

impulse noise level별 이미지

level 5

level 6

impulse noise level별 이미지

level 7

level 8

impulse noise level별 이미지

level 9

level 10
psnr : 30.994

impulse noise inference

	loss	acc
level 1 (1)	0.1731	0.949
level 2 (3)	0.2153	0.928
level 3 (5)	0.3078	0.899
level 4 (6)	0.3689	0.882
level 5 (7)	0.4271	0.858
level 6 (9)	0.5252	0.828
level 7 (11)	0.6191	0.821
level 8 (13)	0.7161	0.785
level 9 (15)	0.7679	0.776
level 10 (17)	0.7946	0.773

median blur

[함수 설명] `new_img = cv2.medianBlur(image,3)`

cv2.medianBlur(src, ksize, dst=None) -> dst

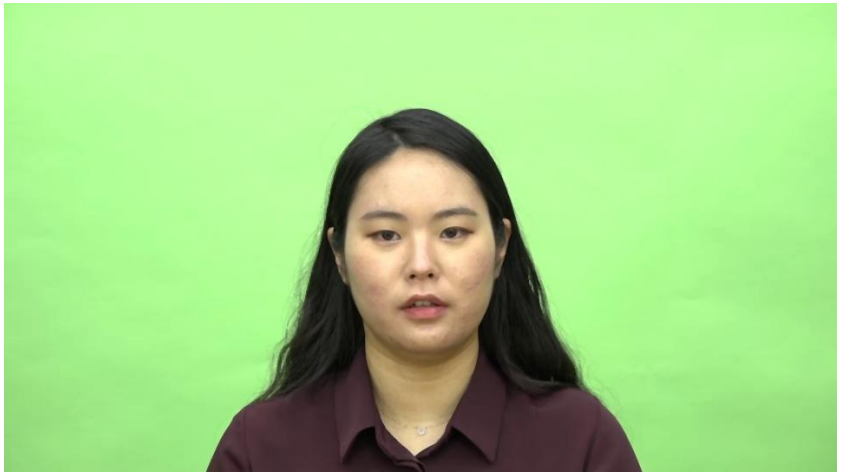
src : 입력 영상. 각 채널 별로 처리됨
ksize : 커널 크기. 1보다 큰 홀수를 지정. 숫자 하나를 집어주면 됨
dst : 출력 영상, src와 같은 크기, 같은 타입

- cv2.medianBlur 함수를 이용하여 미디언 블러 적용
- ksize를 이용하여 강도 조절

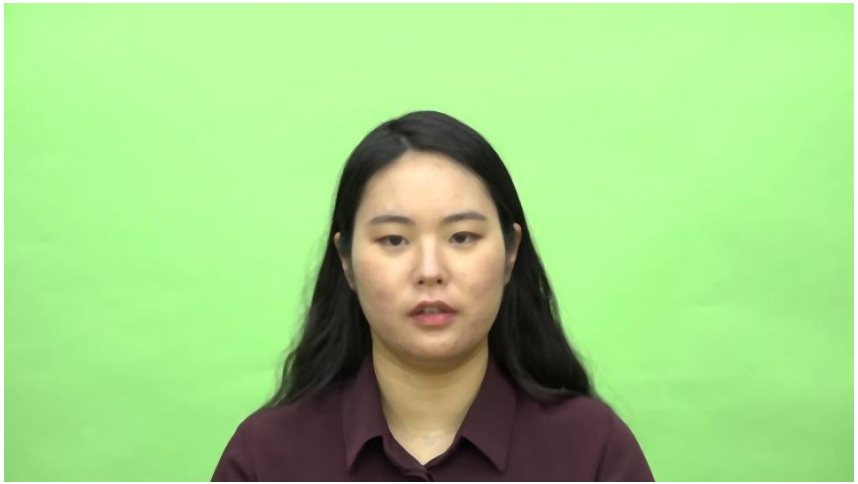
median blur level

level 1	3
level 2	5
level 3	7
level 4	9
level 5	11
level 6	13
level 7	15
level 8	17
level 9	19
level 10	21

median blur level별 이미지

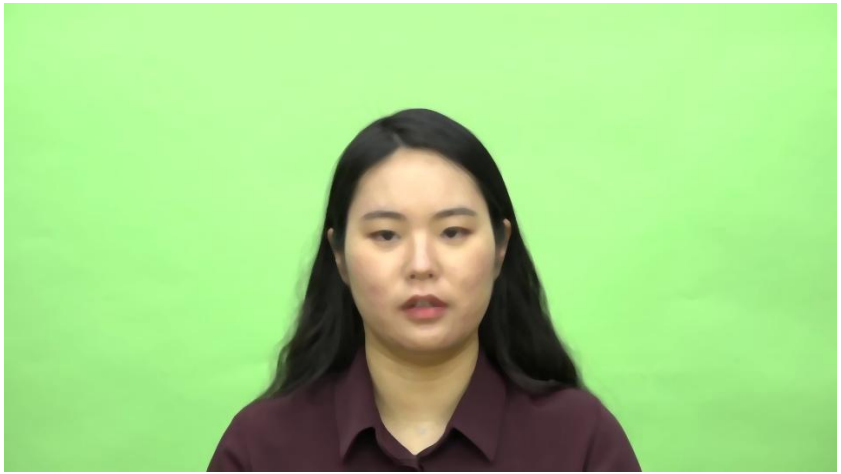


level 1

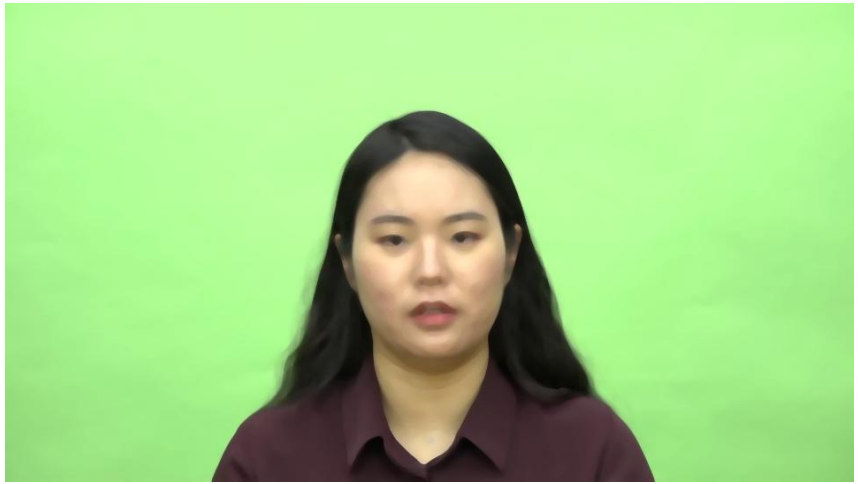


level 2

median blur level별 이미지

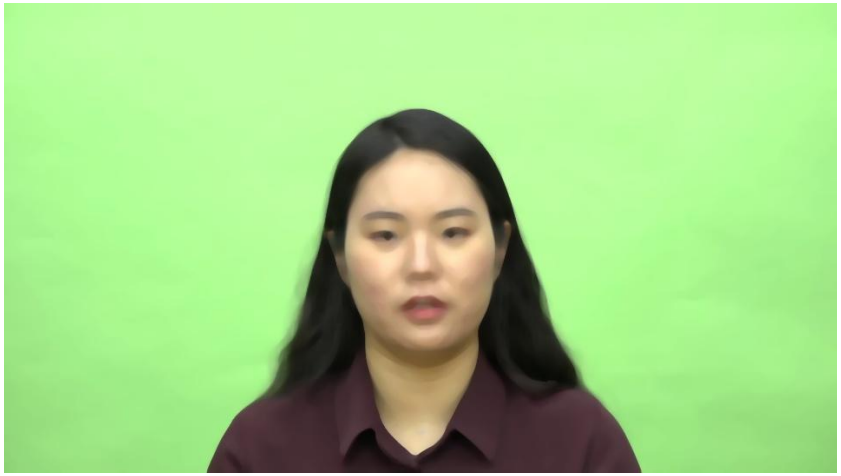


level 3

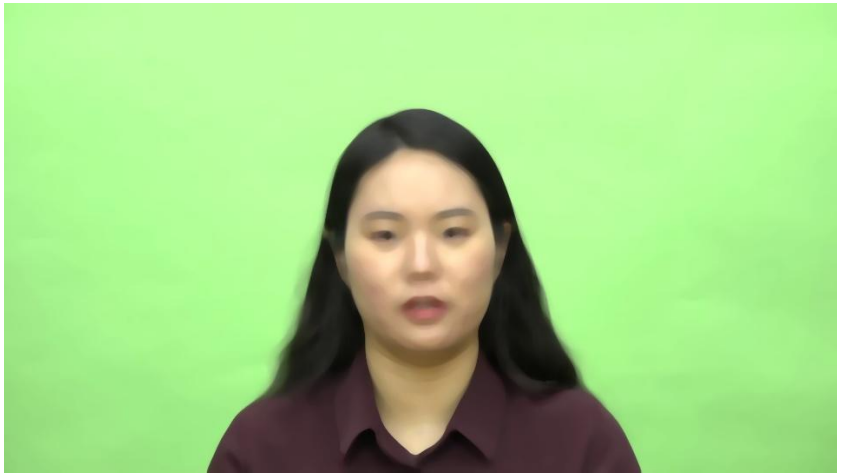


level 4

median blur level별 이미지

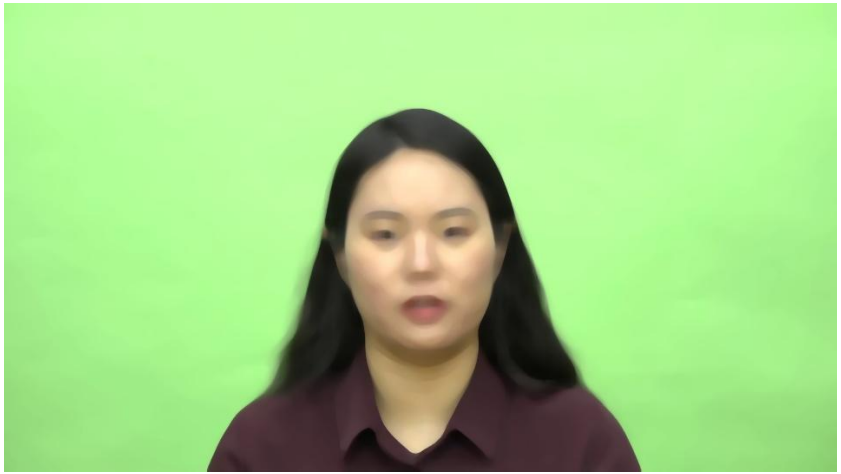


level 5

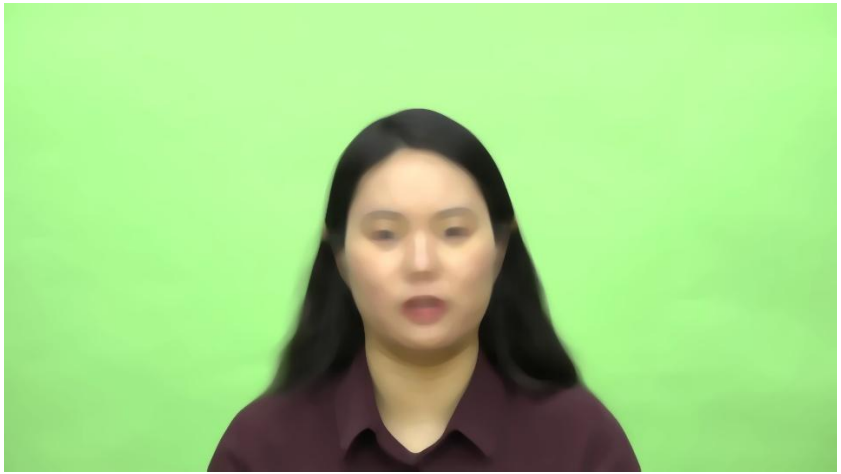


level 6

median blur level별 이미지

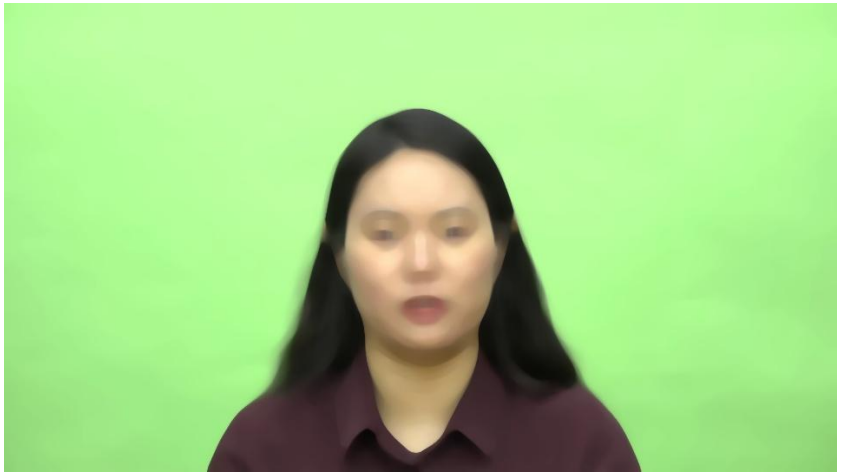


level 7

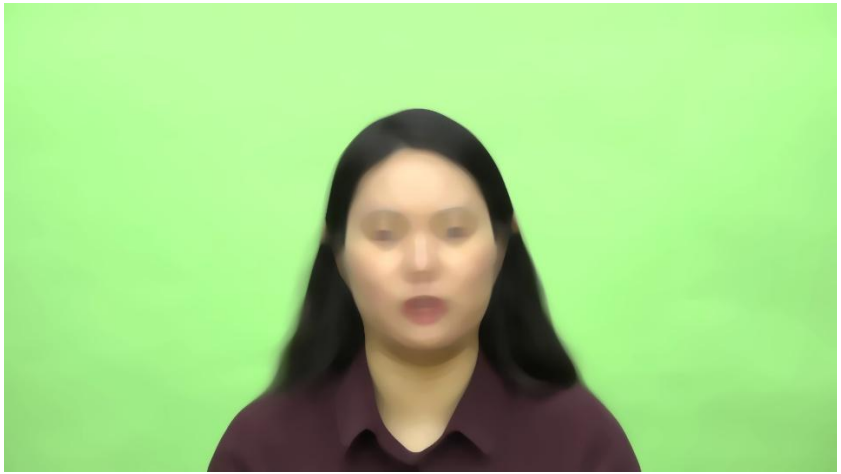


level 8

median blur level별 이미지



level 9



level 10
psnr : 38.7056

median blur inference

	loss	acc
level 1 (3)	0.2335	0.894
level 2 (5)	0.9500	0.706
level 3 (7)	1.8720	0.536
level 4 (9)	2.6689	0.504
level 5 (11)	3.0557	0.5
level 6 (13)	3.2510	0.5
level 7 (15)	3.3703	0.5
level 8 (17)	3.4155	0.5
level 9 (19)	3.5072	0.5
level 10 (21)	3.6061	0.5

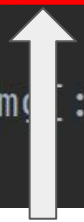
전부 **fake**로 예측

Gaussian Blur

```
path = os.path.join(real_valid, img)
image = cv2.imread(path)

blur_img = cv2.GaussianBlur(image, (i,i), 0)

save_path = real_valid_save_path + img[:-4] + '_lv1_3' + img[-4:]
cv2.imwrite(save_path, blur_img)
```



커널 사이즈 지정으로 강도 조절

Gaussian Blur Level

level 1	3
level 2	5
level 3	7
level 4	9
level 5	11
level 6	13
level 7	15
level 8	17
level 9	19
level 10	21

Gaussian Blur Level 별 이미지

Level1

Level2

Gaussian Blur Level 별 이미지

Level3

Level4

Gaussian Blur Level 별 이미지

Level5

Level6

Gaussian Blur Level 별 이미지

Level7

Level8

Gaussian Blur Level 별 이미지

Level9

Level10
psnr: 35.32

Gaussian Blur Inference

	loss	acc
level 1	1.31	0.573
level 2	2.47	0.514
level 3	3.13	0.507
level 4	3.35	0.503
level 5	3.62	0.500
level 6	3.62	0.500
level 7	3.71	0.500
level 8	3.80	0.500
level 9	3.88	0.500
level 10	3.95	0.500

대부분 fake로 판별하는 경향
-> 영상 합성 시 생기는 blurry한 부분으로 fake라고 판단하는데, real 영상에도 blur가 추가됨으로써 real 영상 또한 fake라고 판단하는 것 같다.

Sharpening - unsharp mask 사용

1) unsharp mask를 사용해 blurry한 image 생성
2) 그 다음, original image에서 blurry 이미지를 빼서 sharp한 부분 검출
3) original image에 sharp한 부분을 더하여 선명한 이미지 생성

```
def unsharp_mask(image, kernel_size=(5, 5), sigma=1.0, amount=1.0, threshold=0):
    """Return a sharpened version of the image, using an unsharp ma      """
    blurred = cv.GaussianBlur(image, kernel_size, sigma)
    sharpened = float(amount + 1) * image - float(amount) * blurred
    sharpened = np.maximum(sharpened, np.zeros(sharpened.shape))
    sharpened = np.minimum(sharpened, 255 * np.ones(sharpened.shape))
    sharpened = sharpened.round().astype(np.uint8)
    if threshold > 0:
        low_contrast_mask = np.absolute(image - blurred) < threshold
        np.copyto(sharpened, image, where=low_contrast_mask)
    return sharpened
```

사용하여
강도 조절

Sharpening - unsharp mask Level

level 1	2
level 2	4
level 3	6
level 4	8
level 5	10
level 6	12
level 7	14
level 8	16
level 9	18
level 10	20

Sharpening - unsharp mask 이미지

Level1

Level2

Sharpening - unsharp mask 이미지

Level3

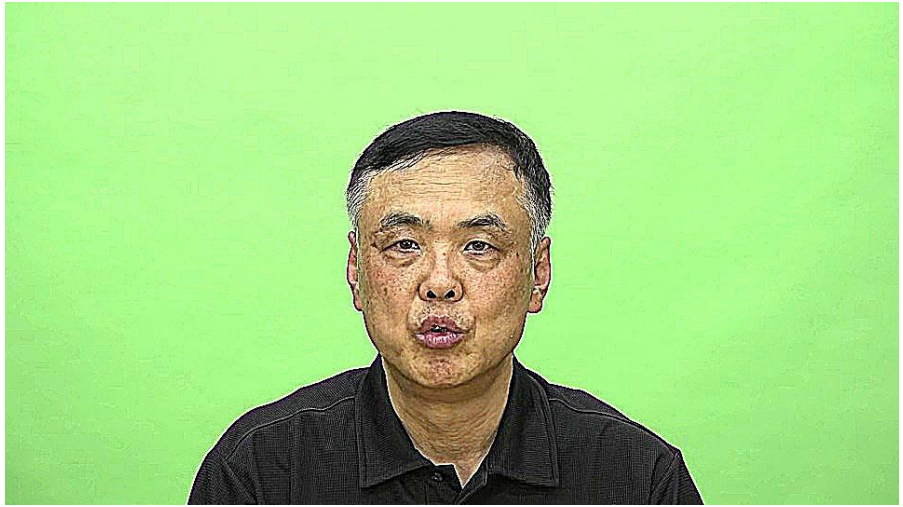
Level4

Sharpening - unsharp mask 이미지

Level5

Level6

Sharpening - unsharp mask 이미지

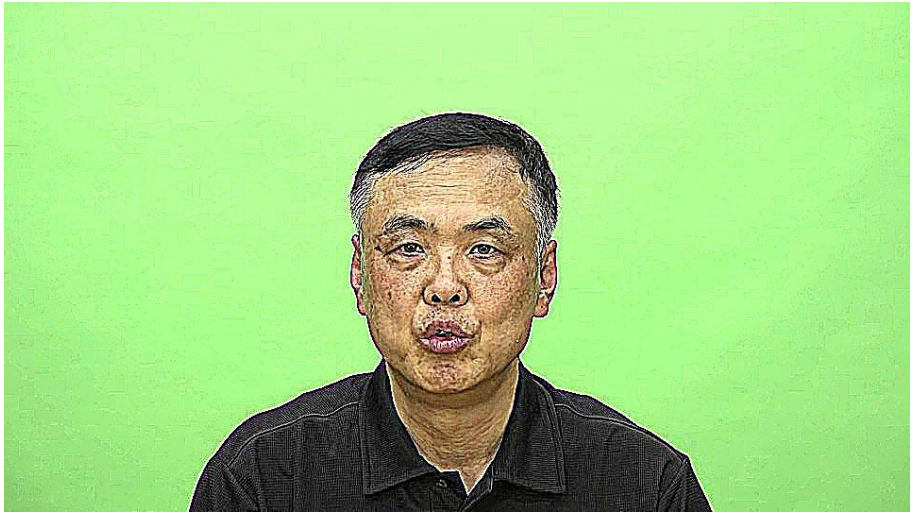


Level7

Level8

Sharpening - unsharp mask 이미지

Level9



Level10

Sharpening - unsharp mask Inference

	loss	acc
level 1	0.525	0.868
level 2	0.493	0.872
level 3	0.333	0.901
level 4	0.230	0.935
level 5	0.168	0.956
level 6	0.137	0.968
level 7	0.134	0.969
level 8	0.155	0.941
level 9	0.191	0.916
level 10	0.236	0.887

bilateral filter

=> 이미지 생성에 너무 많은 시간이 소요되어 이는 나중에 다시 해 볼 예정

```
blur_img = cv2.bilateralFilter(img, 250, 75, 75)
```

```
PSNR : 35.80123545019854
```

250으로 설정한 이미지

=> 500장 생성에 5시간 소요.