캡스톤 디자인

‘딥페이크 탐지’

#11. Noise(Gaussian + Sharpening + JPEG)

김지수, 김민지, 민지민

지난 캡스톤 회의 내용

<코드 수정 및 실험>

- 전처리 단계에서 왜곡이 반영되도록 코드 수정 → 하나의 이미지에 여러 개의 공격을 가하도록 (**공격의 세기 및 공격의 순서가 random 하게 적용**될 수 있도록)

- 학습할 때는 PSNR 30 이상인 공격 파라미터를 랜덤으로 선택하도록 하고, test 시에는 공격 파라미터 weak ~ strong까지 10개 레벨로 구성해서 실험 진행

< '한국정보처리학회 춘계학술대회' 논문 작성>

- 3p로 논문 작성(저번주까지 진행한 내용으로)

❋ 논문 및 코드에 들어갈 공격 : gaussian noise, sharpening, jpeg

공격 함수 · 클래스

```python
def get_attack_params(attack_name):
    attack_dict = {
        'gaussian_noise': {'noise_std': (0, 300)},
        'sharpening' : {'center' : (1, 5)},
        'jpeg' : {'quality' : (5, 95)}
        # 'sp_noise': {'noise_amount': (0.0, 0.01)},
    }

    return attack_dict[attack_name.lower()]
```

PSNR 30 이상이 되도록 공격범위 설정

```python
def fetch_attack():

    attacks = [GaussianNoise.apply, Sharpening.apply]
    random.shuffle(attacks)

    return attacks
```

공격 순서가 랜덤하게 적용되도록 설정

공격 함수 · 클래스

```python
class Sharpening(Function):

    def forward(ctx, image):
        (start, end) = get_attack_params('sharpening')['center']
        i = np.random.uniform(start, end)
        sharpening_arr = np.array([[0, -i, 0],
                                   [-i, 4*i+1, -i],
                                   [0, -i, 0]])
        output = cv2.filter2D(image, -1, sharpening_arr)
        return output
```

sharpening class 정의

```python
class GaussianNoise(Function):

    def forward(ctx,image):
        (start, end) = get_attack_params('gaussian_noise')['noise_std']
        noise_std = np.random.uniform(start, end)
        mean = 0
        sigma = noise_std ** 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 class 정의

공격 함수 · 클래스

```
for attack in attacks:
    train_fakenoise = attack(train_fakenoise)
# JPEG 적용
(start, end) = get_attack_params('jpeg')['quality']
noise_amount = np.random.uniform(start, end)
save_path = save_fakenoise_train_path + fake_img
cv2.imwrite(save_path, train_fakenoise, [cv2.IMWRITE_JPEG_QUALITY, noise_amount])
```

sharpening과 gaussian noise가 랜덤하게 적용된 이미지에 마지막으로 jpeg 적용

noise 이미지 test

```
 1 print('-' * 50)
 2 acc = validate(valid_loader, model, criterion)
```

```
--------------------------------------------------
Valid: 100%|██████████| 1800/1800 [08:14<00:00,  3.64it/s, loss - 5.3672, acc - 0.502]
```

valid list : realnoise valid(900), fakenoise valid(900)에 대해서 test

adversarial train

```
----------
Epoch 1/3
Train:   0%|          | 0/313 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/
  cpuset_checked))
Train: 100%|██████████| 313/313 [11:07<00:00,  2.13s/it, loss - 0.1489, acc - 0.939]
Valid: 100%|██████████| 113/113 [01:36<00:00,  1.17it/s, loss - 0.6761, acc - 0.804]
Epoch 2/3
Train: 100%|██████████| 313/313 [10:34<00:00,  2.03s/it, loss - 0.0437, acc - 0.985]
Valid: 100%|██████████| 113/113 [01:22<00:00,  1.37it/s, loss - 0.1739, acc - 0.941]
Epoch 3/3
Train: 100%|██████████| 313/313 [10:33<00:00,  2.02s/it, loss - 0.0319, acc - 0.989]
Valid: 100%|██████████| 113/113 [01:22<00:00,  1.36it/s, loss - 0.3671, acc - 0.907]
```

train list : 원본 real train(2500), 원본 fake train(2500), realnoise train(2500), fakenoise train(2500)

valid list : 원본 real valid(900), 원본 fake valid(900), realnoise valid(900), fakenoise valid(900)

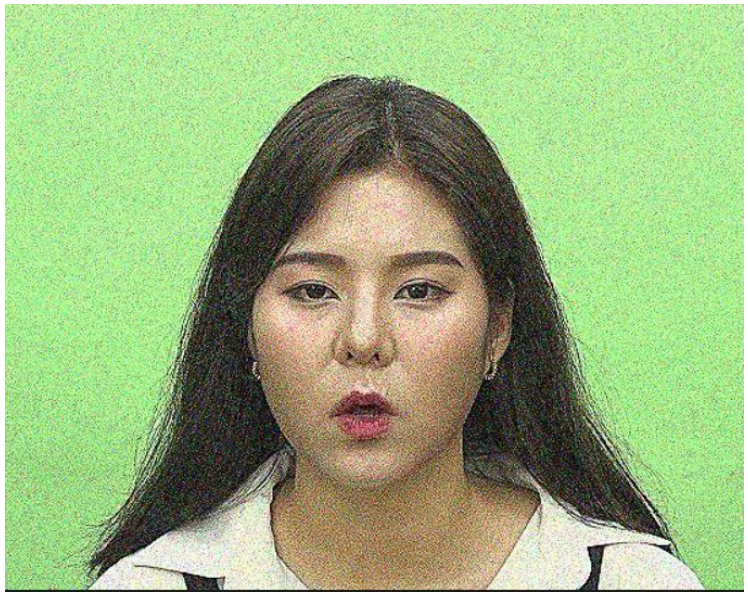
<실험 세팅>	gaussian noise	sharpening	jpeg
LV1(weak)	30	3	90
LV2	60	5	85
LV3	90	7	75
LV4	120	9	65
LV5	150	11	55
LV6	180	13	45
LV7	210	15	35
LV8	240	17	25
LV9	270	19	15
LV10(strong)	300	21	5

test 이미지 noise 적용

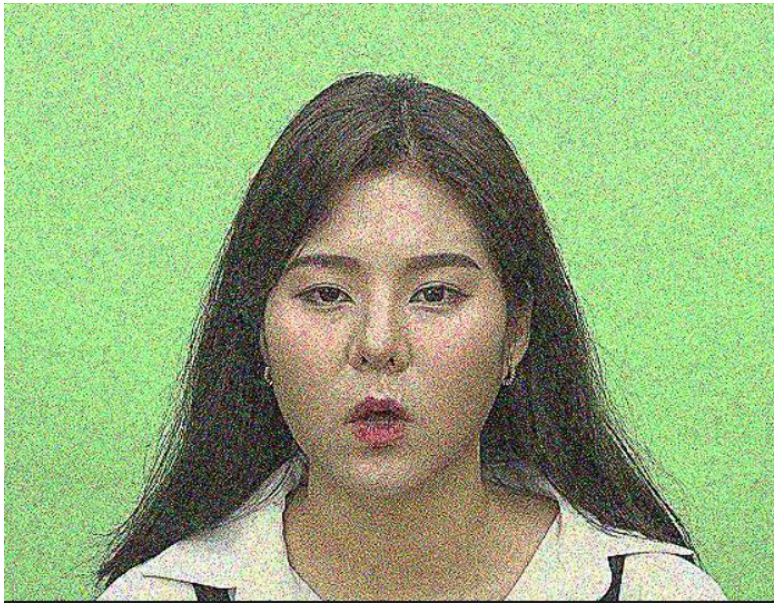
Lv1

Lv2

test 이미지 noise 적용



Lv3



Lv4

test 이미지 noise 적용

Lv5

Lv6

test 이미지 noise 적용

Lv7

Lv8

test 이미지 noise 적용

Lv8

Lv8

test 이미지 noise 적용

Lv9

Lv10

noise test : real noise test + fake noise test

	loss	acc
원본 test 이미지	0.0243	**0.988**
LV1(weak)	0.3342	**0.926**
LV2	0.2384	**0.938**
LV3	0.1938	**0.945**
LV4	0.2510	**0.923**
LV5	0.3331	**0.901**
LV6	0.5952	**0.866**
LV7	0.5447	**0.866**
LV8	0.5269/0.3436	**0.863/0.890**
LV9	0.3219	**0.897**
LV10(strong)	0.2820	**0.922**