

캡스톤 디자인 ‘딥페이크 탐지’

#5. 딥페이크 탐지기 및 안티포렌식 코드 조사

김지수, 김민지, 민지민

지난 캡스톤 회의 내용 - 딥페이크 생성 코드 조사 -

코랩을 이용하여 딥페이크 생성 관련 코드를 실행하여 딥페이크 생성

DeepFaceLab 프로그램을 활용하여 딥페이크를 생성

SimSwap을 이용한 딥페이크 **face swap** 이미지 생성

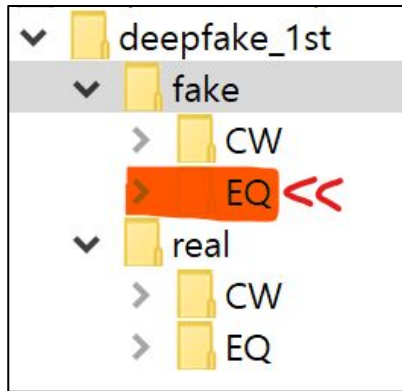
⇒ 이번주 목표

- 딥페이크 탐지기 코드 찾아보고 실행해보기
- 안티 포렌식 생성 코드 찾아보고 실행해보기

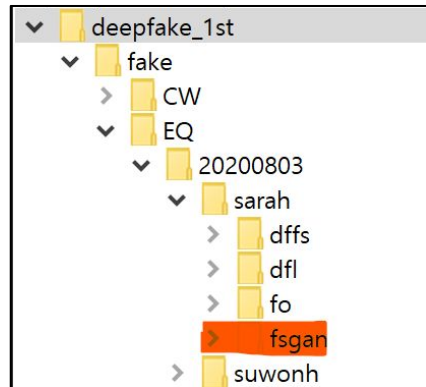
Anti-Forensic 데이터셋 생성

- AIHub 딥페이크 변조 영상 □ 비디오 데이터
- Dacon 딥페이크 변조 영상 탐지 □ 이미지 데이터 (사용)
 - AIHub의 데이터 캡처본

Anti-Forensic 데이터셋 생성

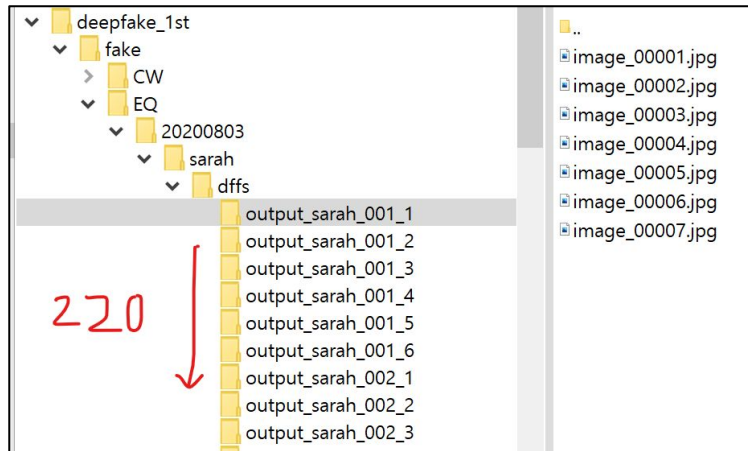
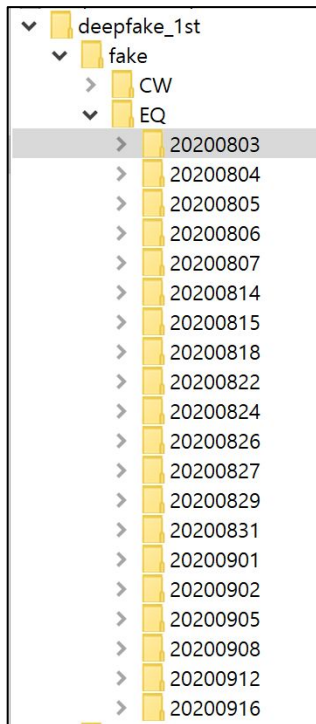


해당 경진대회 주최측의
공지에 따르면,
CW카테고리에 real
데이터가 상당 수 분포
□ 사용X



Fsgan얼굴 생성 방식에서
간혹 real이미지 삽입 되는
문제
□ 사용X

Anti-Forensic 데이터셋 생성



총 데이터셋 126GB

Fake 데이터셋 중 EQ 폴더만 20GB (약 30만장)
EQ폴더 내부에서 이미지 선택 (fsgan폴더 제외)

Anti-Forensic 데이터셋 생성

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
```

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



Anti-Forensic 데이터셋 생성

Denoising

```
dst = cv2.fastNlMeansDenoisingColored(image_RGB, None, 5, 10, 7, 21)
```



Anti-Forensic 데이터셋 생성

JPEG Compression

```
image.save('E:/AntiForensic/jpeg/'+ i, "JPEG", quality = 40)
```



데이콘 - 딥페이크 변조 영상 탐지 AI 경진대회

딥페이크 변조 영상 탐지 AI 경진대회

서울대 | 영상 | GAN | 분류 | Accuracy

₩ 상금 : 1,000만원

🕒 2020.10.19 ~ 2020.11.19 17:59

+ Google Calendar

👤 455명 📅 마감



<https://dacon.io/competitions/official/235655/codeshare>

데이콘 - 딥페이크 변조 영상 탐지 AI 경진대회

1. Xception 모델 사용
2. FaceForensics++ 데이터로 학습된 pre-trained model weight 사용
(출처 - <https://github.com/HongguLiu/Deepfake-Detection>)
1. CenterCropping 사용
: dlib을 이용하여 얼굴만 Cropping하여 학습하였을 때 보다 얼굴 추출없이 CenterCrop하여 학습하였을 때, 결과가 잘나와서 CenterCropping을 사용
1. overfit을 막기 위해 1번만 학습

데이콘 - 딥페이크 변조 영상 탐지 AI 경진대회

```
real/image_000011109.jpg 0 fake/image_000011109.jpg 1
real/image_000011108.jpg 0 fake/image_000011108.jpg 1
real/image_000011107.jpg 0 fake/image_000011107.jpg 1
real/image_000011106.jpg 0 fake/image_000011106.jpg 1
real/image_000011105.jpg 0 fake/image_000011105.jpg 1
real/image_000011079.jpg 0 fake/image_000011079.jpg 1
real/image_000011078.jpg 0 fake/image_000011078.jpg 1
real/image_000011077.jpg 0 fake/image_000011077.jpg 1
real/image_000011076.jpg 0 fake/image_000011076.jpg 1
real/image_000011075.jpg 0 fake/image_000011075.jpg 1
real/image_000011049.jpg 0 fake/image_000011049.jpg 1
real/image_000011048.jpg 0 fake/image_000011048.jpg 1
real/image_000011047.jpg 0 fake/image_000011047.jpg 1
real/image_000011046.jpg 0 fake/image_000011046.jpg 1
real/image_000011045.jpg 0 fake/image_000011045.jpg 1
real/image_000011005.jpg 0 fake/image_000011005.jpg 1
real/image_000011004.jpg 0 fake/image_000011004.jpg 1
real/image_000011003.jpg 0 fake/image_000011003.jpg 1
real/image_000011002.jpg 0 fake/image_000011002.jpg 1
real/image_000011001.jpg 0 fake/image_000011001.jpg 1
real/image_00001985.jpg 0 fake/image_00001985.jpg 1
```

real, fake 각각 100장씩 선택.

120 : train

80 : valid

train / test 임의로 나눔

fake

```
Epoch 1/1
Train: 0%|          | 0/120 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning:
  cpuset_checked))
Train: 100%|██████████| 120/120 [00:26<00:00, 4.49it/s, loss - 0.7753, acc - 0.425]
Valid: 100%|██████████| 80/80 [00:04<00:00, 16.20it/s, loss - 1.2749, acc - 0.512]
```

jpeg

```
Epoch 1/1
Train: 0%|          | 0/120 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481:
  cpuset_checked))
Train: 100%|██████████| 120/120 [00:18<00:00, 6.40it/s, loss - 0.7514, acc - 0.483]
Valid: 100%|██████████| 80/80 [00:11<00:00, 7.13it/s, loss - 3.6464, acc - 0.425]
```

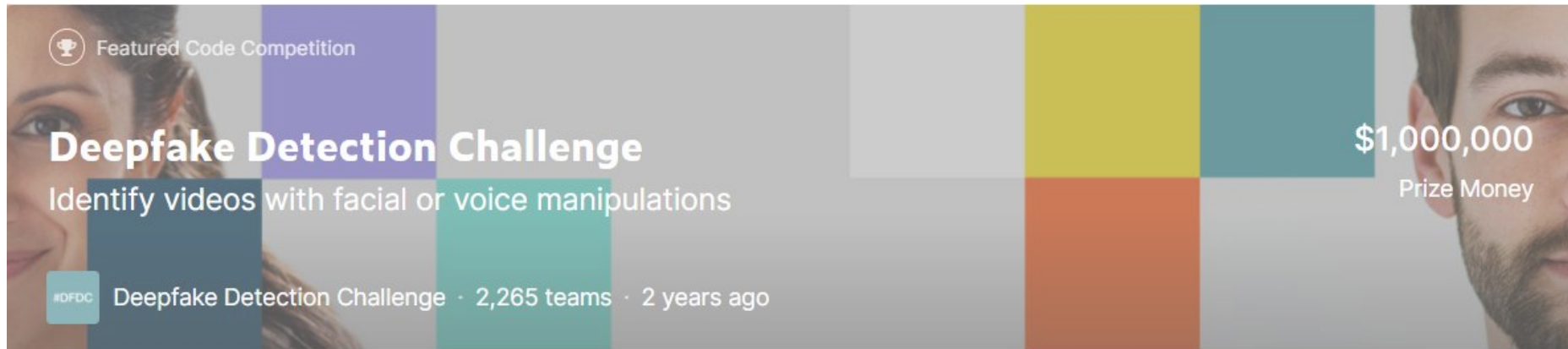
denoise

```
Epoch 1/1
Train: 0%|          | 0/120 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481:
  cpuset_checked))
Train: 100%|██████████| 120/120 [00:25<00:00, 4.72it/s, loss - 0.7639, acc - 0.525]
Valid: 100%|██████████| 80/80 [00:10<00:00, 7.80it/s, loss - 8.7330, acc - 0.450]
```

sharpening

```
Epoch 1/1
Train: 0%|          | 0/120 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481:
  cpuset_checked))
Train: 100%|██████████| 120/120 [00:12<00:00, 9.90it/s, loss - 0.7760, acc - 0.550]
Valid: 100%|██████████| 80/80 [00:10<00:00, 7.61it/s, loss - 0.8099, acc - 0.325]
```

캐글 Deepfake Detection Challenge



<https://www.kaggle.com/robikscube/faceforensics-baseline-dlib-no-internet>

FaceForensics++: Learning to Detect Manipulated Facial Images

FaceForensics++: Learning to Detect Manipulated Facial Images

발행일 : 2019년 1월 25일

Andreas Rössler¹

Davide Cozzolino²

Luisa Verdoliva²

Christian Riess³

발행된 저널 : ICCV 2019

Justus Thies¹

Matthias Nießner¹

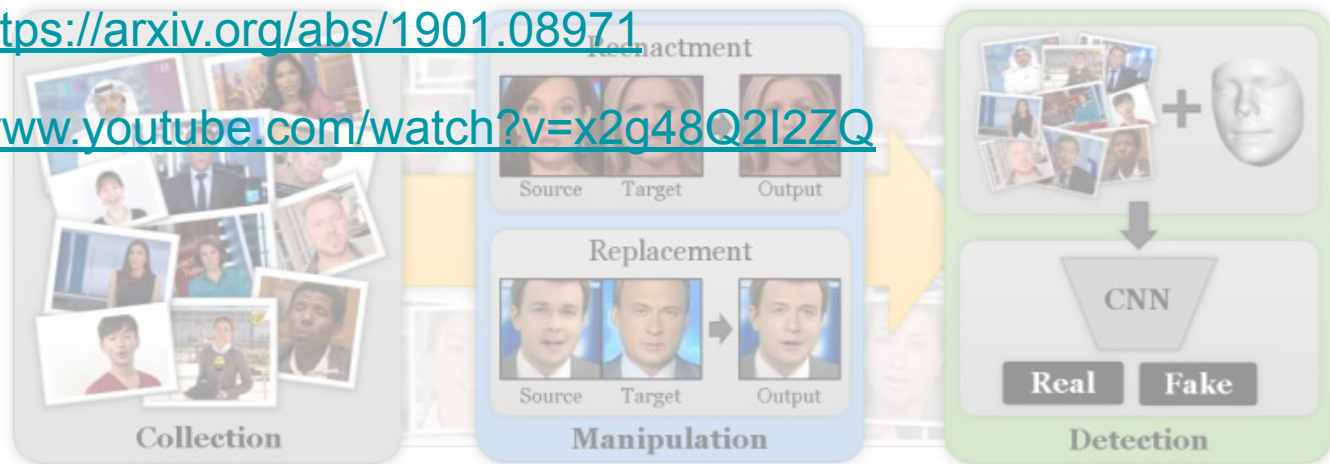
¹Technical University of Munich

²University Federico II of Naples

³University of Erlangen-Nuremberg

링크 : <https://arxiv.org/abs/1901.08971>

<https://www.youtube.com/watch?v=x2g48Q2I2ZQ>



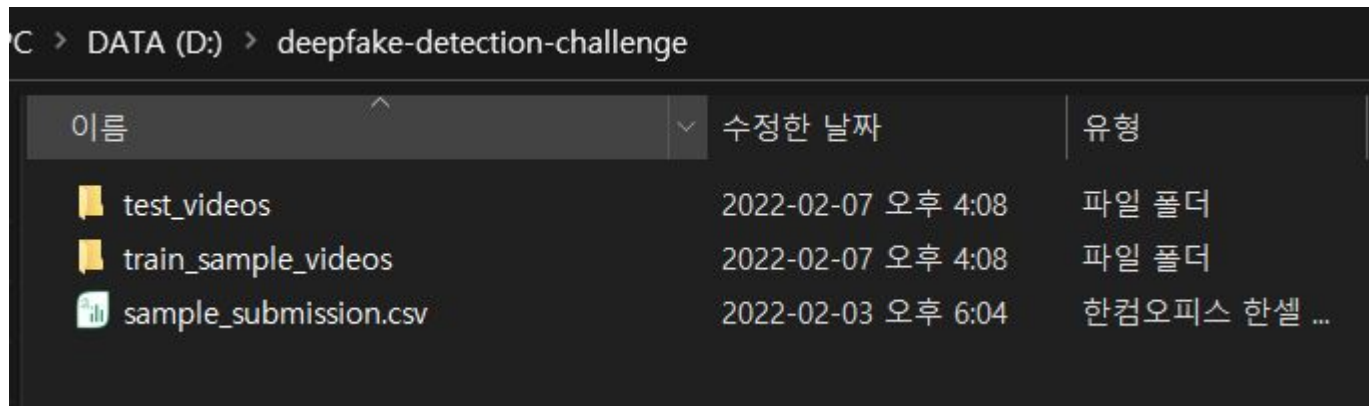
캐글 Deepfake Detection Challenge

test_videos : 400개 동영상

train_sample_videos : 400개 동영상과 metadata.json 파일

metadata.json에서

```
{"aagfhgtpmv.mp4":{"label":"FAKE","split":"train","original":"vudstovrck.mp4"}}
```



이름	수정한 날짜	유형
test_videos	2022-02-07 오후 4:08	파일 폴더
train_sample_videos	2022-02-07 오후 4:08	파일 폴더
sample_submission.csv	2022-02-03 오후 6:04	한컴오피스 한셀 ...

캐글 Deepfake Detection Challenge

dlib 설치 (dlib란 C++로 작성된 툴킷이지만, HOG(Histogram of Oriented Gradients) 특성을 사용하여 얼굴 검출하는 기능이 많이 사용된다.)

FaceForensics++에서 사전 학습된 모델 파일 로드

다양한 유형의 모델을 제공한다.

예측 함수 생성은 비디오 파일이 제공되며 `test_full_image_network` 실행

```
test_full_image_network(video_path, model, output_path, start_frame=0, end_frame=30, cuda=False)
```

출력 avi 파일 저장

<https://github.com/ondyari/FaceForensics>

캐글 Deepfake Detection Challenge

전체 이미지 모델 테스트 -> 전체_raw.p, 전체_c40.p, 전체_c23.p

face_detection/xception 모델 테스트 중 -> xception_all_raw.p, xception_all_c40.p, xception_all_c23.p

열차 세트에 대한 예측 검증: 50개의 프레임을 가짜인지 진짜인지 예측한다.

샘플링된 모든 프레임의 최대, 최소 및 평균 "가짜" 예측을 반환합니다.

얼굴을 탐지할 수 없으면 0.5 예측

face_detection/xception/all_c23.p 모델이 좋은 결과로 보여 deepfake detection 모델로 최종 사용

캐글 Deepfake Detection Challenge

<All_c23 모델로 판별한 영상의 예>

TRUE LABEL인 영상을 진짜로 판별하는 모습을 볼 수 있다.

그에 반해, FAKE LABEL인 영상을 가짜로 판별하는 건 단 2프레임 밖에 없다.

