B반 1조 Faceswap과 GAN을 이용한 신변보호 서비스

2021.10

김미향 서정빈 임영선 채다미 박진우 임태미 박규석

B반 1조 Faceswap과 GAN을 이용한 신변보호 서비스

B반 1조 Faceswap과 GAN을 이용한 신변보호 서비스

1. 프로젝트 개요

1.1. 조원 구성 및 역할

- '그것이 모르겠다'에서는 총 7명의 팀원을 영상팀 3명 음성팀 4명으로 나눠서 프로젝트를 진행함. 특정 역할을 개인에게 분배하기보다 모두가 개발에 참여해 ai 관련 경험을 쌓게 끔 함.

[그림] 단체 사진

- 영상팀(서정빈, 임영선 박진우)과 음성팀(박규석, 김미향, 채다미, 임태미)으로 구분하여 기술을 구현하였다.

1.2. 프로젝트 선정 배경 및 목표

- 국내 방송사의 경우 모자이크와 음성 변조를 통해 초상권을 보호하고 있다. 하지만, 모자이크의 빈도가 해외의 다른 방송사에 비해 높아 영상 시청의 몰입감을 저해한다는 의견이 많으며, 간단한 검색을 통해 손쉽게 구할 수 있는 모자이크 제거 프로그램은 본연의 목적인 초상권의 보호를 위협하고 있다. 이에 가상의 얼굴과 목소리를 통해 모자이크의 활용 빈도를 줄이면서도 영상 속 등장인물의 초상권을 보호하고자 한다.

- 해당 프로젝트에 있어 핵심적으로 사용되는 기술인 얼굴 생성과 합성, 목소리 변조 등은

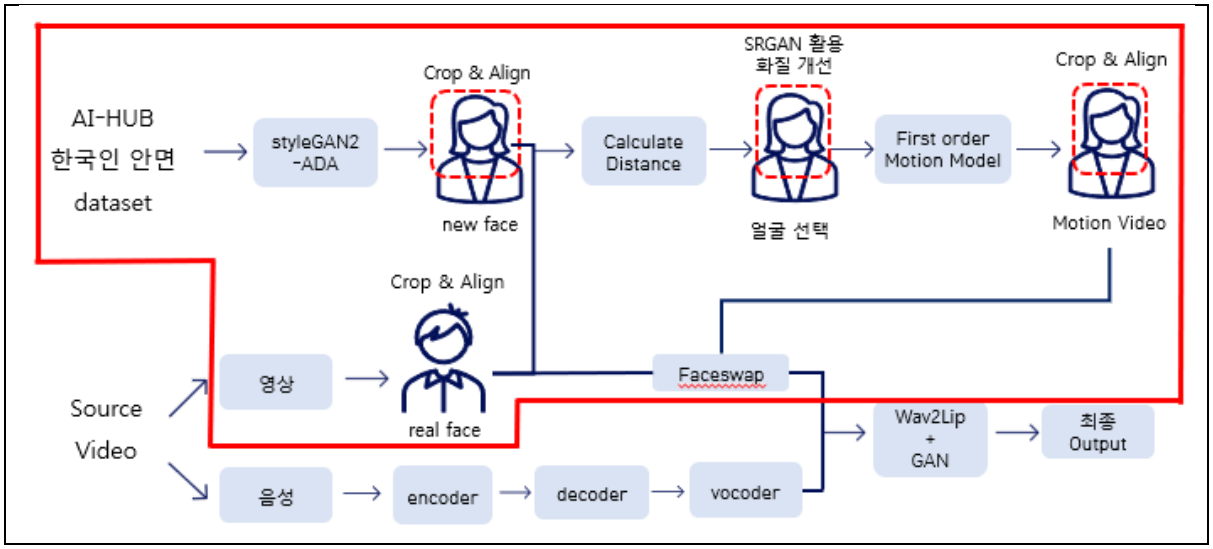
현재 많은 기업에서 관심을 가지고 있으며, 사회적으로 이슈가 되고 있는 분야이다. Virtual Human Rui의 경우 실제 촬영한 영상에 가상의 얼굴을 합성하는 딥페이크 기술을 기반으로 하고 있으며, 클로바에서는 HDTS(Hybrid DNN Text-to-Speech) 기술을 활용하여 오디오북에 연예인의 목소리를 추가하는 상품을 출시하였다. 추가적으로 해당 기술에 대한 연구도 활발히 이루어지고 있어 해당 프로젝트를 최종적으로 선정하였다.

- 본 프로젝트는 가상의 얼굴과 목소리의 생성을 통한 발전된 신변보호를 목표로 한다. 또한, 현재 딥페이크와 딥보이스의 분야는 악질적인 합성으로 기술이 가지고 있는 발전 가능성보다 악용의 우려가 높다. 해당 프로젝트를 통해 딥페이크와 딥보이스를 피해자의 신변 보호라는 공익적 목적으로 활용하고자 한다.

2. 영상파트

2.1. 시스템 구조 및 흐름

[그림] 영상파트 시스템 구조도



- 영상 파트에서는 AI-hub 의 한국인 안면 데이터셋을 전처리 한 후 stylegan2-ada 를 통해 가상 얼굴을 생성함. 이후, 소스 비디오에서 추출한 영상의 실제 얼굴과 가상 얼굴의 유사도를 계산하여 생성된 가상 얼굴 중 가장 다른 얼굴을 선정함. First order motion model 을 활용하여 해당 가상 얼굴을 기반으로 생성한 동영상과 기존의 소스 비디오를 인풋으로 하여 faceswap 을 진행함.

2.2. 데이터셋

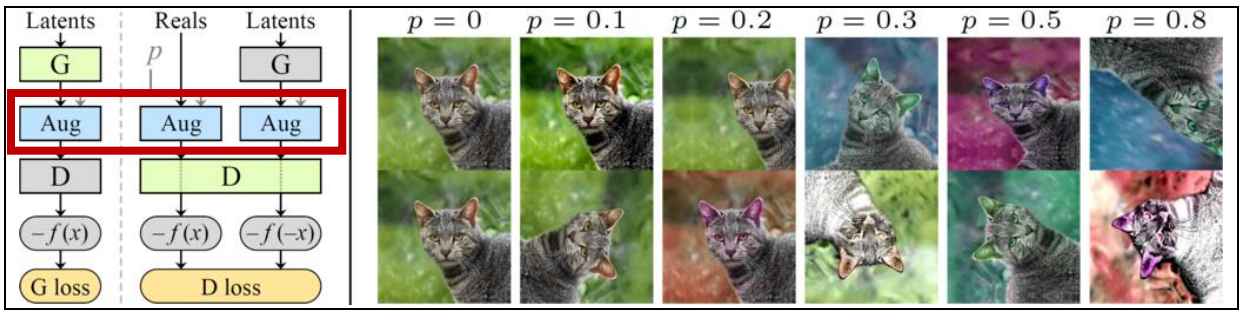
[그림] AI-hub 데이터 구조

- 한국인에 맞는 새로운 얼굴을 생성해주기 위해서 AIhub 의 한국인 안면 이미지 데이터를 활용함. 배경과 인물의 구분이 뚜렷한 밝은 조명의 사진과, 안경, 선글라스, 모자 등 액세서리를 착용하지 않은 정면 사진 데이터를 학습에 사용하기 위해 전처리를 진행함.
- 전체 폴더에서 악세서리를 착용하지 않은 S001, 가장 빛이 밝은 L1, 카메라를 정면으로 바라보는 C7 순으로 폴더 안에 있는 데이터를 활용함. 결과적으로 400 장의 정면사진을 데이터셋으로 활용함.

2.3. StyleGan2 – ada

- 기존의 stylegan2 의 특징으로는 고해상도 이미지를 생성가능하며, initial layer 에서 low resolution 을 점진적으로 증가시킨다. 그러나, 학습을 시키기 위해 많은 이미지 데이터를 필요로 한다. 이는 discriminator 로 하여금 overfitting 될 위험이 있다. 이에 adaptive discriminator augmentation 을 활용하여 개수가 적은 제한된 데이터셋을 통해서 학습을 안정화시킴.

[그림] stylegan2-ada



- 가상환경 생성과 필요 패키지 설치

```
Requirements
64-bit Python 3.7 and PyTorch 1.7.1.
CUDA toolkit 11.0 or later.
```

새로운 StyleGan2 라는 가상환경 만들어주고 pytoch 1.7.1, cuda 11.0 download

```
conda create -n StyleGan2 python=3.7 jupyter notebook
conda activate StyleGan2
conda install pytorch==1.7.1 torchvision==0.8.2 torchaudio==0.7.2
cudatoolkit=11.0 -c pytorch
git clone https://github.com/NVlabs/stylegan2-ada-pytorch.git
python -m ipykernel install --user --name StyleGan2 --display-name
"StyleGan2"
import torch
print(torch.__version__)
```

파이토치 1.7.1 버전 확인

B반 1조 Faceswap과 GAN을 이용한 신변보호 서비스

- 학습을 위해 얼굴을 기준으로 정방형으로 데이터를 전처리 해줌.

```
python dataset_tool.py --source ~/데이터셋 경로 --dest ~/저장될 경로
```

아래에 있는 Transform option을 통해 사진을 자를 방식을 width, height option을 통해 높이와 너비 지정가능 함. 해당 코드를 통해 -dest에 지정된 경로에 png 파일 생성됨.

[그림 1] dataset_tool.py option

```
@click.command()
@click.pass_context
@click.option('--source', help='Directory or archive name for input dataset', required=True, metavar='PATH')
@click.option('--dest', help='Output directory or archive name for output dataset', required=True, metavar='PATH')
@click.option('--max-images', help='Output only up to `max-images` images', type=int, default=None)
@click.option('--resize-filter', help='Filter to use when resizing images for output resolution', type=click.Choice(['box', 'lanczos']), default='lanczos', show_default=Tru
@click.option('--transform', help='Input crop/resize mode', type=click.Choice(['center-crop', 'center-crop-wide']))
@click.option('--width', help='Output width', type=int)
@click.option('--height', help='Output height', type=int)
```

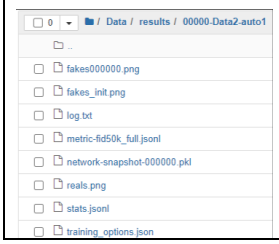
- 학습 과정은 다음과 같음.

```
python train.py --data ~/저장될 경로 --outdir ~/저장될 경로
```

사용할 gpu 개수, train 횟수, tick 횟수 등 지정 가능함. 학습이 진행중이라면 아래와 같이 확인할 수 있음.

```
tick 0     kimg 0.0      time 41s          sec/tick 3.5     sec/kimg 109.43  maintenance 37.7   cpumem 3.47   gpumem 6.28   augment 0.000
Evaluating metrics...
{"results": {"fid50k_full": 333.3107447637283}, "metric": "fid50k_full", "total_time": 318.2006196975708, "total_time_str": "5m 18s", "num
.pkl", "timestamp": 1632561148.2050698}
tick 1     kimg 4.0      time 7m 24s       sec/tick 80.1    sec/kimg 20.02   maintenance 322.3  cpumem 3.79   gpumem 6.22   augment 0.006
tick 2     kimg 8.0      time 8m 45s       sec/tick 81.1    sec/kimg 20.27   maintenance 0.1    cpumem 3.79   gpumem 6.22   augment 0.012
tick 3     kimg 12.0     time 10m 06s      sec/tick 81.0    sec/kimg 20.25   maintenance 0.1    cpumem 3.79   gpumem 6.22   augment 0.019
tick 4     kimg 16.0     time 11m 27s      sec/tick 81.2    sec/kimg 20.30   maintenance 0.0    cpumem 3.79   gpumem 6.22   augment 0.024
```

outdir 살펴보면 아래와 같음. Tick 을 default 값으로 주면 매 50 tick 마다 generator 가 생성한 예시를 제공함. 첫번째 진행한 fakes000000.png 의 경우 다음과 같음. 또한, pkl 파일도 생기는데 이 파일을 활용해 새로운 얼굴 사진 생성 가능함. 학습이 진행됨에 따라 지정한 tick 에 따라 generator 에서 생성한 얼굴을 확인할 수 있음.



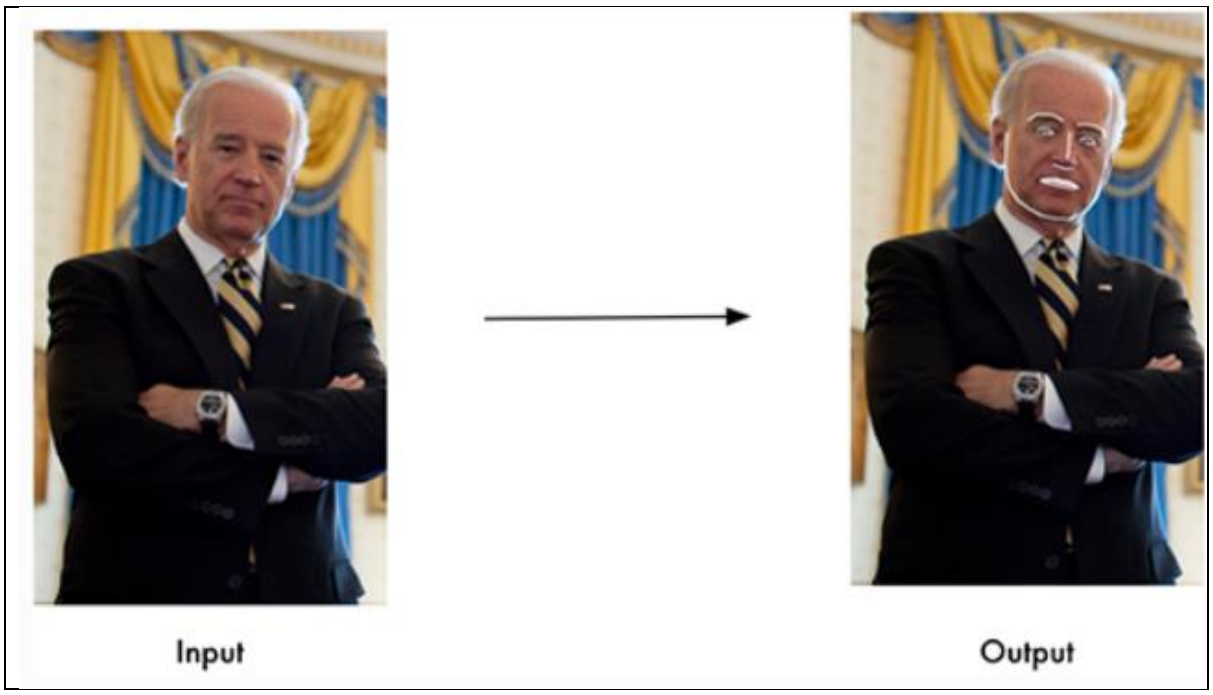
- 가상 얼굴 생성 과정은 다음과 같음.

```
python generate.py --outdir ~/저장될 경로 --trunc=원하는 값 --seeds= 랜덤 시드 \ --network ~/활용할 pkl 파일경로 지정
```

2.4. Face Recognition

- 프로젝트 수행에 있어 제한적인 학습 시간을 이유로 생성한 얼굴 중 가장 사람같은 이미지를 선별해 줄 필요가 있었음. 또한, 프로젝트 목적이 신변보호이므로 기존의 타겟 얼굴과 가장 닮지 않은 얼굴을 Face recognition 를 활용하여 선별함

[그림] Face Recognition 예시



- 코드 진행은 다음과 같음.

1. 활용하기 위해 해당 패키지 설치

pip install face_recognition

2. 얼굴간 비교를 위해 얼굴 영역 박스를 기준으로 잘라내기

```
def get_cropped_face(image_file):

    image = face_recognition.load_image_file(image_file)    # 이미지 불러오기
    face_locations = face_recognition.face_locations(image)   # 얼굴 영역 박스
    a, b, c, d = face_locations[0]     # 얼굴 영역 박스 좌표
    cropped_face = image[a:c,d:b,:]    # 얼굴 영역 박스 좌표를 이용해 얼굴 잘라내기
    return cropped_face
```

3. 잘라낸 얼굴을 활용하여 타겟과 생성 얼굴의 특징점의 값을 가지고 있는 128 차원의 embedding vector 를 추출함

```
def get_face_embedding(face):
    return face_recognition.face_encodings(face)
```

```
def get_face_embedding_dict(dir_path):
    file_list = os.listdir(dir_path)
    embedding_dict = {}

    for file in file_list:
        img_path = os.path.join(dir_path, file) # 경로를 병합하여 새 경로 생성
        try:
            face = get_cropped_face(img_path)    # 얼굴 영역만 자른 이미지
        except:                                  # 인식하지 못하는 이미지는 넘어감
            continue

        embedding = get_face_embedding(face)   # 얼굴 영역에서 얼굴 임베딩 벡터를 추출
        if len(embedding) > 0:   # 얼굴 영역이 제대로 detect되지 않았을 경우를 대비
                # os.path.splitext(file)[0]에는 이미지파일명에서 확장자를 제거한 이름이 담긴다.
            embedding_dict[os.path.splitext(file)[0]] = embedding[0]

    return embedding_dict
```

```
target = get_face_embedding_dict(_path + "target") # 타겟의 정면사진 한 장 활용
```

```
file = get_face_embedding_dict(_path + "face")
```

4. Euclidean distance 활용하여 가장 다른 얼굴 이미지 구하기

```
def get_distance(name1, name2):
    return np.linalg.norm(target[name1]-file[name2], ord=2)
```

```
def unsimilar_face(file):
    _list = []
    for key in file.keys():
        _list.append((key, get_distance('target', key)))
        _list.sort(key = lambda x: x[1], reverse = True)
    top_5 = _list[0:5]
    return(top_5)
```

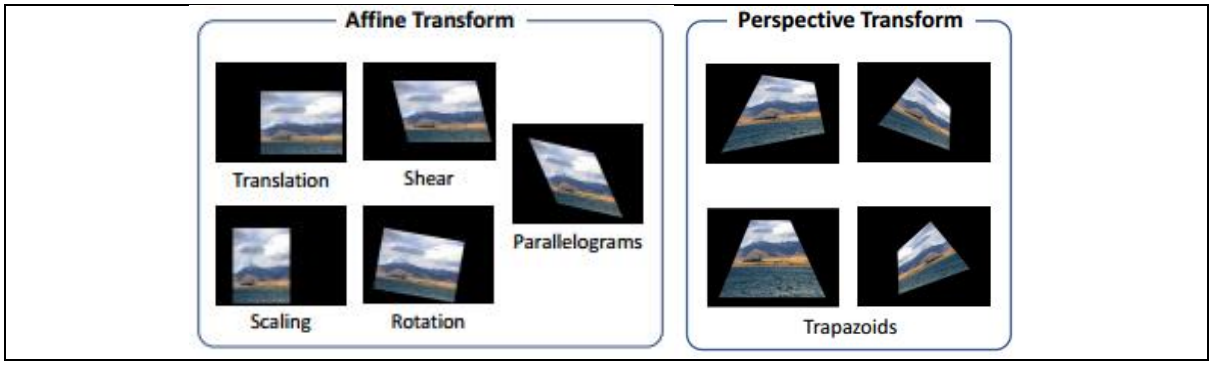
```
unsimilar_face(file)
```

5. 최종적으로 5 개의 제일 다른 얼굴 중 faceswap 에 활용할 생성 얼굴 선택

2.5. Affine Transformation & Perspective Transformation

- Stylegan2-ada 를 활용하여 새로운 얼굴을 생성하였으나, 사용하는 사진은 1 장으로 faceswap 을 진행해주기에 데이터 개수가 부족함. 따라서, 이 한 장을 가지고 여러 구도의 사진을 만들 수 있는 방안으로 Affine Transformation & Perspective Transformation 을 시도함.

[그림] Face Recognition 예시



- Affine Transform 의 경우 translation, shear, scaling, rotation, paralleogram 등 2 차원에서의 변환이며, Perspective Transformation 의 경우 Affine Transform 과 비교하였을 때, 차원이 더 높음. 직사각형보다 자유도가 높은 사다리꼴, 임의의 사각형으로 표현하는 3 차원에서의 원근법 원리를 적용한 변환임.

- 활용 코드는 다음과 같음.

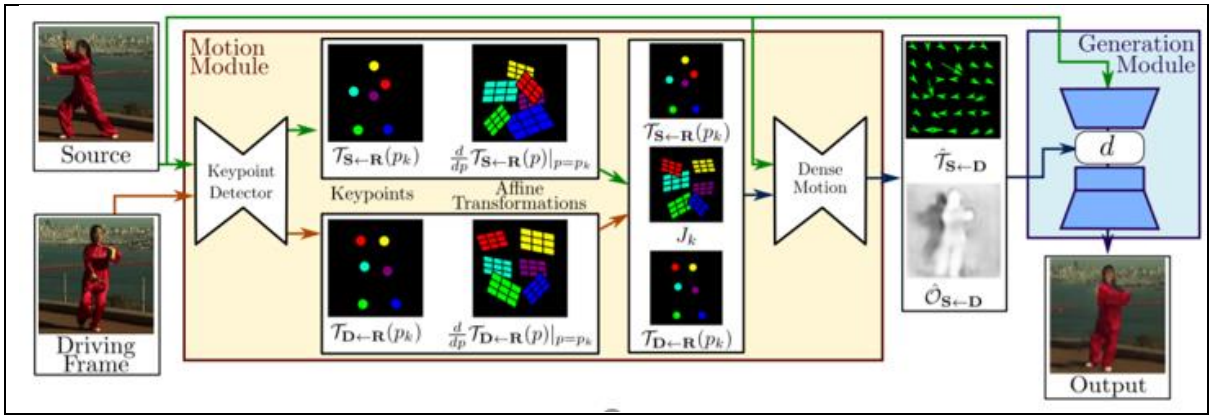
```
file_name = "./증명사진.png"
img = cv2.imread(file_name,1)
rows, cols = img.shape[:2]  # 맨 마지막은 channel 수이다.

#---① 원근 변환 전후 4개 좌표
pts1 = np.float32([[0,0], [0,rows], [cols, 0], [cols,rows]])  # pts1: 변환 이전 영상의 좌표 4개, 4 x 2 배열
pts2 = np.float32([[200,200], [200,rows-200], [cols,0], [cols,rows]])   # pts2: 변환 이후 영상의 좌표 4개, 4 x 2

#---② 변환 전 좌표를 원본 이미지에 표시
cv2.circle(img, (0,0), 10, (255,0,0), -1)
cv2.circle(img, (0,rows), 10, (0,255,0), -1)
cv2.circle(img, (cols,0), 10, (0,0,255), -1)
cv2.circle(img, (cols,rows), 10, (0,255,255), -1)

#---③ 원근 변환 행렬 계산
mtrx = cv2.getPerspectiveTransform(pts1, pts2)  # mtrx: 변환행렬 반환, 3 x 3 행렬
#---④ 원근 변환 적용
dst = cv2.warpPerspective(img, mtrx, (cols, rows))
```

- 얼굴 데이터의 측면, 입의 움직임 등 다양한 표현을 하는데 제한적이였음. 또한, perspective transformation 으로 인해 얼굴 크기가 작아져, 이후 얼굴크기를 정면 사진과 비슷하게 나오도록 resize 를 해야하는 추가적인 작업이 발생하여 해당 방식으로 data augmentation 을 진행하지 않고 first-order-motion-model 활용함.

2.6. First Order Motion Model

- 사진을 통해 다양한 상황을 표현함과 동시에 faceswap 에 활용하려면 영상의 형태로 만들어 줄 필요가 있었음. 따라서, 소스 이미지와 비디오를 통해 소스 이미지를 바탕으로 한 비디오 시퀀스를 생성할 수 있는 first order motion model 을 활용함.

[그림] First Order Motion Model 구조



- Source : 소스 이미지의 경우 animate 해주기 위한 사진으로 moving video 가 필요하지 않음.
- Driving Frame : 모션이 있는 소스와 유사한 개체의 비디오 시퀀스를 가져옴. 해당 Driving Frame 의 모션을 소스 이미지에 포함시키기 위해 사용함.
- Motion Module : Source 와 Driving Frame 을 입력으로 받아 로컬 아핀 변환과 오클루전 마스크를 출력으로 내는 단계. Keypoint Detector 은 unsupervised detector 로 source 와 driving frame 의 키포인트를 예측함. 로컬 아핀 변환은 각 keypoint 주변에서의 변환을 수행하며, 이를 통해 직선, 길이의 비, 평행선을 보존함.
- Dense Motion Network : keypoints 와 transformations Tx:S from R, Tx:D from R 을 입력으로 받아 dense motion field 와 occlusion mask 를 출력함.
- Generator Network : Source image 의 appearance 와 Dense Motion 으로 표현된 모션을 사용, 보이지 않는 소스의 객체 부분을 생성해줌.
- 깃허브에서 colab 으로 구현햔 demo.ipynb 활용함. 아래와 같이 stylegan2 를 활용하여 생성한 얼굴을 source 로 주었고 측면, 눈 깜빡임, 입의 움직임을 주기 위해 Driving Frame 으로 직접 촬영한 영상을 활용하였음. 결과적으로 다음과 같은 영상을 얻을 수 있었음.

[그림] First Order Motion Model 결과

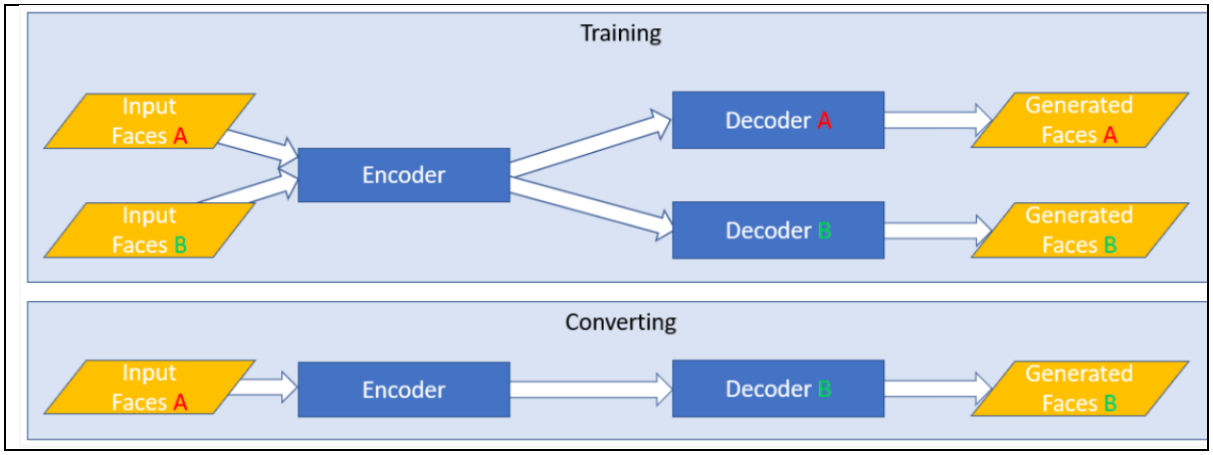
생성 얼굴	측면	눈 움직임	입 움직임
			

2.7. FaceSwap

- Faceswap 에 활용할 알고리즘으로 GAN 을 기반으로 한 model 과 Auto-encoder 모델을 고려하였음. GAN 기반 모델의 경우 학습의 시간이 오래걸리며, 결과물이 좋지 못하였으나, Auto-encoder 기반 모델의 경우 비교적 적은 학습 시간으로도 준수한 결과물을 뽑을 수 있었음.

B반 1조 Faceswap과 GAN을 이용한 신변보호 서비스

[그림] Faceswap AutoEncoder 구조



- 학습에서 두 영상을 Shared Encoder 로 활용하고 서로 다른 Decoder 로 이용함. 이를 통해 두 얼굴에서 공통적인 특징들을 추출하고 두 얼굴에서 서로 다른 특징을 Decoder 에서 구분하여 이미지를 복원함. 학습함.
- 바꾸고 싶은 영상의 이미지를 Encoder 에 넣어서 추출된 얼굴의 특징점을 Decoder B 에 넣어주면 해당 위치에 맞게 B 의 이미지 인물이 가지는 특징이 입혀지게 되며, faceswap 을 해줌.
- 활용 방법은 다음과 같음.

1. 환경 세팅

```
conda create -n forswap python=3.8 jupyter notebook
conda activate forswap
python -m ipykernel install --user --name forswap --display -name "forswap"
git clone https://github.com/deepfakes/faceswap.git
cd faceswap
python setup.py
```

2. Mp4 -> png 로 변환

추출하고 싶은 동영상의 디렉토리 안의 mp4 파일 지정해준 후 output dir 로 설정해줌. 해당 코드를 통해 동영상을 프레임 단위로 추출됨.

```
python tools.py effmpeg -a extract -i source_input_dir -o source_output_dir
python tools.py effmpeg -a extract -i target_input_dir -o target_output_dir
```

3. Extract – 학습에 필요한 데이터셋으로 변환

프레임 단위로 추출된 사진에서 mtcnn 을 통해 얼굴을 인식 후 정방형 사이즈로 이미지를 변환함.

```
python faceswap.py extract -i 2 의 source_output_dir -o source_data
python faceswap.py extract -i 2 의 target_output_dir -o target_data
```

4. Train - 3 에서 만든 2 개의 데이터셋을 활용해서 학습 후 model_save_dir 에 저장됨. default 값으로 지정할 경우 25000 iteration 마다 checkpoint 생성되며, 진행상황 확인할 수 있음.

```
python faceswap.py train -A source_data_dir -B target_data_dir -m model_save_dir
```

5. Convert – extract 하기 전의 raw data dir 로 지정해주며, convert 된 사진이 convert_dir 에 저장됨.

-c 옵션을 통해 avg_color, color_transfer, manual-balance, match-hist, seamless-clone 등 지정 가능함.

```
python faceswap.py convert -i source_output_dir -o convert_dir -m model_save_dir
```

6. Png -> Mp4

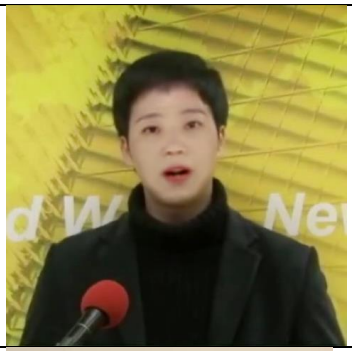
-a gen-vid => png 파일을 다시 영상으로 만들어 줌.

-r => reference video 주소 지정해주면 됨.

```
python tools.py effmpeg -a gen-vid -i convert_dir -r source_input_dir
```

- Faceswap 의 결과는 그림과 같음.

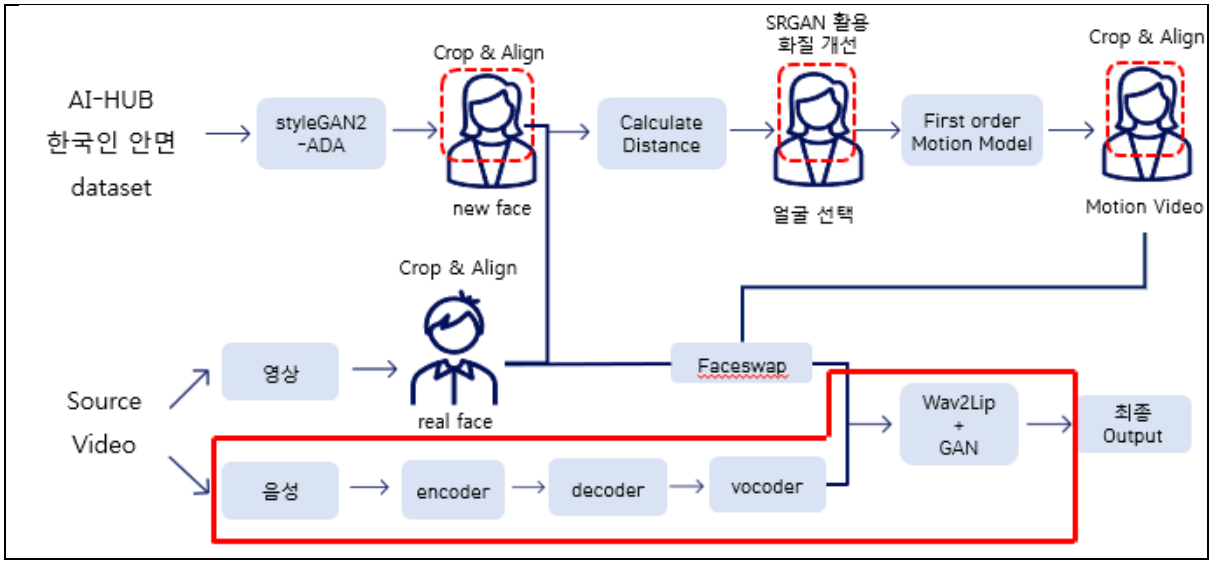
[그림] Faceswap 결과

A	B	swap
		
		

3. 영상파트

3.1. 시스템 구조 및 흐름

[그림] 영상파트 시스템 구조



- 원본 영상에서 영상과 음성을 분리한 후, 음성을 컴퓨터가 인식할 수 있는 숫자로 변경하는 인코더 과정, 숫자를 멜스펙트로그램으로 변경하는 디코더, 멜스펙트로그램을 다시 음성으로 변경하는 보코더 과정을 통해 새로운 목소리의 음성을 생성함.
- Wav2Lip과 gan 방법을 이용해 새로운 음성과 새로운 얼굴의 영상을 합쳐 최종 아웃풋을 얻어냅니다.

3.2. 자료 조사 및 기술 선정 과정

- 현재 음성 변조의 문제점: 기존의 음성변조는 실제 음성에 음성의 파동 변화를 주어 변조를 하는 것이기 때문에 목소리 변조의 위험성이 존재함. 음성 변화와 관련하여 다음과 같은 자료를 조사함
- Voice Conversion(VC) : 음성 변환은 문자를 음성으로 변환하거나 음성 특성들을 변화시키거나 음성, 감정, 악세트를 인식하는 음성 합성 분야의 일부라고 할 수 있음(Sisman, Yamagishi, King & Li, 2020). 음성 변환은 Speech Analysis, Spectral Conversion, Prosody Conversion, Speaker Characterization, Vocoding과 같은 다양한 음성 처리 기술들을 총칭함.
- Voice Conversion(VC)의 Deep learning 기술 동향 중 가장 많이 언급되는 Encoder-

Decoder with Attention 기술은 기존의 Encoder-Decoder은 짧은 단어들은 학습이 되지만 긴 문장일수록 제대로 학습하지 못하는 한계에서 발전한 모형임. 이 때, Encoder는 쉽게 말해 텍스트를 숫자로 변환하는 과정, Decoder는 해당 음성을 시각화하는 Mel spectrogram을 생성하며, Recursive Neural Network를 사용하며 이전 단어와 현재 단어 사이의 관계를 기반으로 미래 단계를 예측할 때 주로 사용. Vocoder는 Mel Spectrogram을 음성으로 만들어주는 과정을 말함. 이 때, Attention은 사람 뇌의 기능을 바탕으로 본 떠 만든 것으로 몇 정보를 무시하지만 중요한 정보들에 가중치를 두어 중요 정보들을 추출하는 과정임(Sisman et al., 2020).

- 본 프로젝트에서는 음성 변조가 필요한 피해자의 인터뷰 영상에서 음성을 추출 후, 배경음과 실제 음성을 분리하고, 이를 텍스트로 추출하여 새로운 음성을 입혀 가상의 얼굴로 입힌 영상과 합성하는 작업을 하고자 함.

- 해당 프로젝트를 수행하기 위해 관련한 기술로 Voice Style Transfer, Voice Conversion, Tacotron 2, STT(Speech to Text), TTS(Text to Speech), Deep fake speech의 오픈 소스 코드를 활용함.

4. 기술 선정 과정 및 코드 구현

- 본 프로젝트에서는 영상에서 음성 추출, 해당 음성을 text로 변환, 다른 음성으로 변환 과정을 거치고자 하였으며 단계마다 기술 선정은 다음과 같음

4.1. 영상에서 음성 추출:

- ffmpeg 를 활용하여 비디오에서 wav, mp3 등의 음성 파일을 추출함. Spleeter를 활용하여 인터뷰 영상에서 배경음악과 화자를 분리할 수 있음.

4.2. 음성에서 텍스트 추출:

- 음성 인식(Speech Recognition) 과정인 STT(Speech to text)에서는, 음성을 입력받고 해당하는 문자를 변환하는 기술. 딥러닝을 이용한 STT모델은 WaveNet 등이 존재하며 본 프로젝트에서는 WaveNet의 활용에 실패하여 Speech Recognition을 이용하여 간편하게 구현을 하고자 하였다. 음성 인식에는 학습할 음성 데이터의 부족함(본 데이터는 조원의 1시간 30분 정도의 음성 데이터 정보를 활용함)과 해당 기술 코드가 복잡하여 시간이 오래 걸리기 때문에 SpeechRecognition을 활용함.

- 추가적으로 해당 텍스트의 맞춤법 및 띄어쓰기 확인을 위해 py-hanspell을 설치하여 해당 text 파일의 맞춤법을 추가적으로 검사하여 text 추출의 정확도를 높이고자 하였음.

4.3. 텍스트에서 새로운 음성 추출:

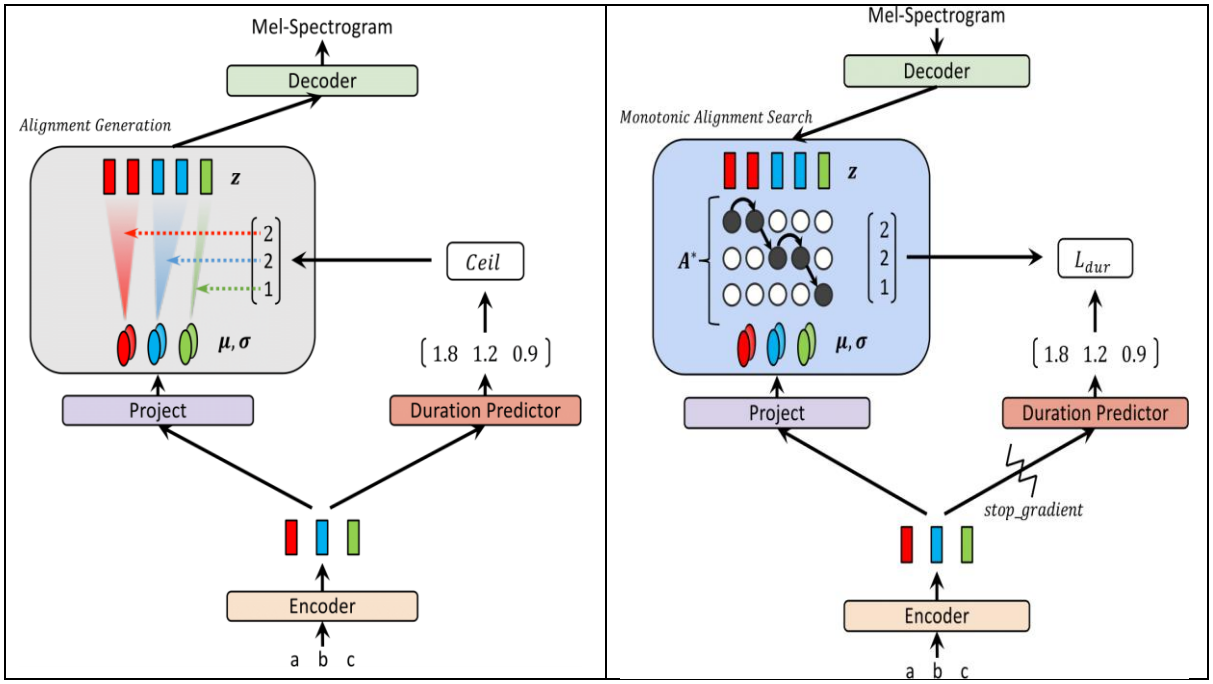
- 음성 합성(Speech Synthesis)의 대표적인 모델로 Tacotron 2가 있으며 Tacotron2의 기본적인 동작 방법은 text로부터 Mel Spectrogram을 생성하며 이 때 생성된 Mel Spectrogram을 Waveglow를 통과시켜 음성을 생성함. 멜 스펙트로그램(Mel Spectrogram)은 음성의 frequency가 Mel Scale로 변환되는 스펙트로그램을 말한다.

- 본 프로젝트에서는 Glow-TTS & Hifi-GAN 모델을 활용하여 TSS를 활용함. 적은 시간으로도 빠르게 학습을 진행할 수 있으며 둘 중 하나의 모델만을 학습시켜도 음성 합성 결과를 볼 수 있기 때문임(Kong, Kim, & Bae, 2020).

[표] Speech Synthesis(TTS) 관련 모델

Model	설명	최종선정
Tacotron2	LSTM encoder decoder based model that generates spectrograms	
GlowTTS	Glow 기반의 spectrogram generator	o
FastSpeech2	Non-autoregressive transformer-based spectrogram generator that predicts duration, energy, and pitch	
FastPitch	Non-autoregressive transformer-based spectrogram generator that predicts duration and pitch	
WaveGlow	Glow 기반의 보코더	
MelGAN	GAN 기반의 보코더	
HifiGAN	GAN 기반의 보코더	o
FastPitch_HIfiGAN_E2E	Beginner에게 추천, FastPitch와 hifiGAN을 합한 End-to-end 모델	
FastSpeech2_HifiGAN_E2E	FastSpeech2와 hifiGAN을 합한 End-to-end 모델	

[그림] Glow_TTS training & inference



-

- 추가적으로 NAVER의 Clova TTS와 Google의 TTS의 API의 활용 가능성을 검토함.₩

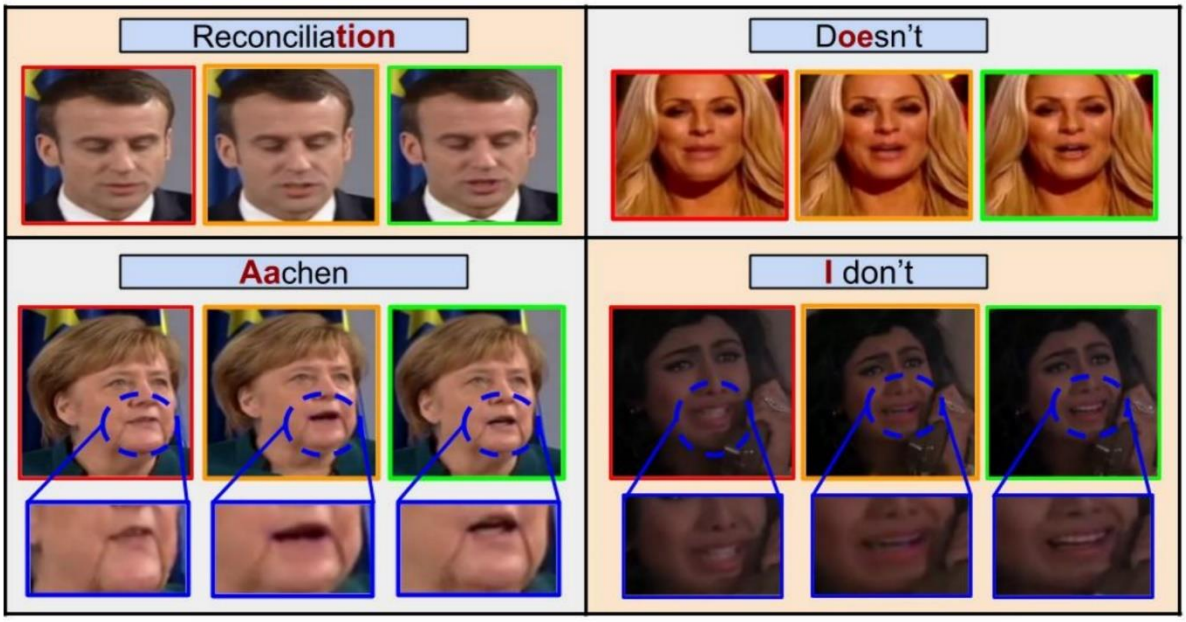
4-4. 영상의 얼굴과 새로운 음성 간 싱크 맞추기

- 본 프로젝트에서는 새로운 음성을 영상에 바로 넣으면 입모양과 나오는 발화 내용이 달라서 해당 음성 싱크 조정 필요하였음. DeepFake와 관련하여 Video generation, talking face generation, lip sync 등의 영상과 음성을 맞출 수 있는 기술을 추가적으로 조사하여 적용하였음

[표] Wav2Lip Model

Model	설명	선정
Wav2Lip	음성과 입술 움직임을 높은 정확도로 싱크를 맞춤	
Wav2Lip + GAN	입술 움직임과 음성의 싱크의 정확도는 wav2lip보다 낮지만 비디오 화질은 좀 더 나음	o

- Wav2Lip + GAN 모델을 활용하여 새롭게 합성된 얼굴 비디오 화질을 유지하고 새로 만든 오디오 간의 입술의 싱크를 맞춤.



[그림] wav2lip 예제. 노란색과 초록색의 그림이 해당 wav2lip으로 구현된 딥페이크 영상임. -tion, oe, Aa, I의 발음에 맞춰서 정교하게 변화하는 것을 볼 수 있음.

5. 소스코드 및 작동 과정

- 음성 처리와 관련한 소스 코드들은 Colab과 google drive를 사용함

5.1. ffmpeg, spleeter

```
!pip install ffmpeg
!pip install spleeter

inputdir = '/content/drive/MyDrive/proj'

for filename in os.listdir(inputdir):
    actual_filename = filename[:-4]
    if(filename.endswith(".mp4")):
        os.system('ffmpeg -i {} -acodec pcm_s16le -
ar 16000 {}.wav'.format(filename, actual_filename))
    else:
        continue
```

다음과 같은 코드로 해당 영상에서 음성 데이터를 추출하거나 영상 구간을 자를 수 있음

```
!ffmpeg -i {영상이름}.mp4 {저장할 음성이름}.wav
```

```
!ffmpeg -i {영상이름}.mp4 -ss 00:00:02 -to 00:00:30 -c copy {저장할 영상 이름}.mp4
```

하단의 코드로 배경음과 화자의 소리를 분리할 수 있음

```
!spleeter separate -o output/ {파일 이름}.wav
```

추출된 음성을 확인

```
fig = plt.figure(figsize=(14, 4))
korean_wav, rate = librosa.core.load('{음성파일}.wav')
librosa.display.waveplot(korean_wav, sr=rate)
ipd.Audio(korean_wav, rate=rate)
```

5.2. Speech Recognition

STT – Speech Recognition 활용

```
!pip install SpeechRecognition

import speech_recognition as sr

r = sr.Recognizer()

r = sr.Recognizer()
with sr.AudioFile('{파일 이름}.wav') as source:
     audio = r.record(source)
text_4 = r.recognize_google(audio, language='ko-KR')
text_4
```

STT - Naver Clova speech to text

```
#네이버 api 신청 후, 돌리시면 됩니당
#네이버 Speech to text 음성인식
import sys
import requests
client_id = "" #이건 각자 자기 id와 pw 로 쓰면 됨! #
client_secret = "" #자신이 받은 네이버 api pw 를 쓰세요! #
lang = "Kor" # 언어 코드 ( Kor, Jpn, Eng, Chn )
url = "https://naveropenapi.apigw.ntruss.com/recog/v1/stt?lang=" +
 lang
data = open('{파일이름}.wav', 'rb')
headers = {
    "X-NCP-APIGW-API-KEY-ID": client_id,
    "X-NCP-APIGW-API-KEY": client_secret,
    "Content-Type": "application/octet-stream"
```

```
}
response = requests.post(url,  data=data, headers=headers)
rescode = response.status_code
print(rescode)
if(rescode == 200):
    print (response.text)
else:
    print("Error : " + response.text)
```

5.3. 한국어 맞춤법 검사

텍스트의 띄어쓰기 및 맞춤법 검사

```
!pip install git+https://github.com/haven-jeon/PyKoSpacing.git

!pip install git+https://github.com/ssut/py-hanspell.git

from pykospacing import spacing

new_sent = sent.replace(" ", '')
print(new_sent)



from hanspell import spell_checker
spelled_sent = spell_checker.check(text_2)
hanspell_sent = spelled_sent.checked
print(hanspell_sent)
```

5.4. TTS(Text to Speech)

Glow-TTS & Hifi-GAN 모델로 음성데이터 파일을 학습시켜 text 를 speech 로 변환.py

```
      import os
      import sys
      from pathlib import Path
```

```
%cd /content
!git clone --depth 1 https://github.com/sce-tts/TTS.git -b sce-tts
!git clone --depth 1 https://github.com/sce-tts/g2pK.git
%cd /content/TTS
!pip install -q --no-cache-dir -e .
%cd /content/g2pK
!pip install -q --no-cache-dir "konlpy" "jamo" "nltk" "python-
mecab-ko"
!pip install -q --no-cache-dir -e .
```

```
%cd /content/g2pK
import g2pk
g2p = g2pk.G2p()
```

```
%cd /content/TTS
import re
import sys
from unicodedata import normalize
import IPython

from TTS.utils.synthesizer import Synthesizer

def normalize_text(text):
    text = text.strip()

    for c in ",;:":
        text = text.replace(c, ".")
    text = remove_duplicated_punctuations(text)

    text = jamo_text(text)

    text = g2p.idioms(text)
    text = g2pk.english.convert_eng(text, g2p.cmu)
    text = g2pk.utils.annotate(text, g2p.mecab)
    text = g2pk.numerals.convert_num(text)
    text = re.sub("/[PJEB]", "", text)

    text = alphabet_text(text)

    # remove unreadable characters
    text = normalize("NFD", text)
    text = "".join(c for c in text if c in symbols)
    text = normalize("NFC", text)

    text = text.strip()
    if len(text) == 0:
        return ""

    # only single punctuation
    if text in '.!?':
        return punctuation_text(text)

    # append punctuation if there is no punctuation at the end of
the text
    if text[-1] not in '.!?':
        text += '.'

    return text
```

```
def remove_duplicated_punctuations(text):
    text = re.sub(r"[.?!]+\?", "?", text)
    text = re.sub(r"[.?!]+!", "!", text)
    text = re.sub(r"[.?!]+\.", ".", text)
    return text


def split_text(text):
    text = remove_duplicated_punctuations(text)

    texts = []
    for subtext in re.findall(r'[^.!?\n]*[.!?\n]', text):
        texts.append(subtext.strip())

    return texts


def alphabet_text(text):
    text = re.sub(r"(a|A)", "에이", text)
    text = re.sub(r"(b|B)", "비", text)
    text = re.sub(r"(c|C)", "씨", text)
    text = re.sub(r"(d|D)", "디", text)
    text = re.sub(r"(e|E)", "이", text)
    text = re.sub(r"(f|F)", "에프", text)
    text = re.sub(r"(g|G)", "쥐", text)
    text = re.sub(r"(h|H)", "에이치", text)
    text = re.sub(r"(i|I)", "아이", text)
    text = re.sub(r"(j|J)", "제이", text)
    text = re.sub(r"(k|K)", "케이", text)
    text = re.sub(r"(l|L)", "엘", text)
    text = re.sub(r"(m|M)", "엠", text)
    text = re.sub(r"(n|N)", "엔", text)
    text = re.sub(r"(o|O)", "오", text)
    text = re.sub(r"(p|P)", "피", text)
    text = re.sub(r"(q|Q)", "큐", text)
    text = re.sub(r"(r|R)", "알", text)
    text = re.sub(r"(s|S)", "에스", text)
    text = re.sub(r"(t|T)", "티", text)
    text = re.sub(r"(u|U)", "유", text)
    text = re.sub(r"(v|V)", "브이", text)
    text = re.sub(r"(w|W)", "더블유", text)
    text = re.sub(r"(x|X)", "엑스", text)
    text = re.sub(r"(y|Y)", "와이", text)
    text = re.sub(r"(z|Z)", "지", text)

    return text
```

```
def punctuation_text(text):
    # 문장부호
    text = re.sub(r"!", "느낌표", text)
    text = re.sub(r"\?", "물음표", text)
    text = re.sub(r"\.", "마침표", text)

    return text


def jamo_text(text):
    # 기본 자모음
    text = re.sub(r"ㄱ", "기역", text)
    text = re.sub(r"ㄴ", "니은", text)
    text = re.sub(r"ㄷ", "디귿", text)
    text = re.sub(r"ㄹ", "리을", text)
    text = re.sub(r"ㅁ", "미음", text)
    text = re.sub(r"ㅂ", "비읍", text)
    text = re.sub(r"ㅅ", "시옷", text)
    text = re.sub(r"ㅇ", "이응", text)
    text = re.sub(r"ㅈ", "지읒", text)
    text = re.sub(r"ㅊ", "치읓", text)
    text = re.sub(r"ㅋ", "키읔", text)
    text = re.sub(r"ㅌ", "티읕", text)
    text = re.sub(r"ㅍ", "피읖", text)
    text = re.sub(r"ㅎ", "히읗", text)
    text = re.sub(r"ㄲ", "쌍기역", text)
    text = re.sub(r"ㄸ", "쌍디귿", text)
    text = re.sub(r"ㅃ", "쌍비읍", text)
    text = re.sub(r"ㅆ", "쌍시옷", text)
    text = re.sub(r"ㅉ", "쌍지읒", text)
    text = re.sub(r"ㄳ", "기역시옷", text)
    text = re.sub(r"ㄵ", "니은지읒", text)
    text = re.sub(r"ㄶ", "니은히읗", text)
    text = re.sub(r"ㄺ", "리을기역", text)
    text = re.sub(r"ㄻ", "리을미음", text)
    text = re.sub(r"ㄼ", "리을비읍", text)
    text = re.sub(r"ㄽ", "리을시옷", text)
    text = re.sub(r"ㄾ", "리을티읕", text)
    text = re.sub(r"ㄿ", "리을피읍", text)
    text = re.sub(r"ㅀ", "리을히읗", text)
    text = re.sub(r"ㅄ", "비읍시옷", text)
    text = re.sub(r"ㅏ", "아", text)
    text = re.sub(r"ㅑ", "야", text)
    text = re.sub(r"ㅓ", "어", text)
    text = re.sub(r"ㅕ", "여", text)
    text = re.sub(r"ㅗ", "오", text)
    text = re.sub(r"ㅛ", "요", text)
    text = re.sub(r"ㅜ", "우", text)
    text = re.sub(r"ㅠ", "유", text)
```

```
    text = re.sub(r"ㅡ", "으", text)
    text = re.sub(r"ㅣ", "이", text)
    text = re.sub(r"ㅐ", "애", text)
    text = re.sub(r"ㅒ", "얘", text)
    text = re.sub(r"ㅔ", "에", text)
    text = re.sub(r"ㅖ", "예", text)
    text = re.sub(r"놔", "와", text)
    text = re.sub(r"내", "왜", text)
    text = re.sub(r"뇌", "외", text)
    text = re.sub(r"눠", "워", text)
    text = re.sub(r"눼", "웨", text)
    text = re.sub(r"뉘", "위", text)
    text = re.sub(r"늬", "의", text)

    return text


def normalize_multiline_text(long_text):
    texts = split_text(long_text)
    normalized_texts = [normalize_text(text).strip() for text in t
exts]
    return [text for text in normalized_texts if len(text) > 0]

def synthesize(text):
    wavs = synthesizer.tts(text, None, None)
    return wavs
```

```
synthesizer = Synthesizer(
    "{glow TTS 학습시킨 모델}.pth.tar",
    "/content/drive/MyDrive/TTS_park/glow_tts/config.json",
    None,
    "{hifiGAN 학습시킨 모델}.pth.tar",
    "/content/drive/MyDrive/TTS_park/hifi_gan/config.json",
    None,
    None,
    False,
)
symbols = synthesizer.tts_config.characters.characters
```

```
wav = synthesizer.tts({변환할 텍스트}, None, None)
IPython.display.display(IPython.display.Audio(wav, rate=22050))
```

5.4. Wav2Lip

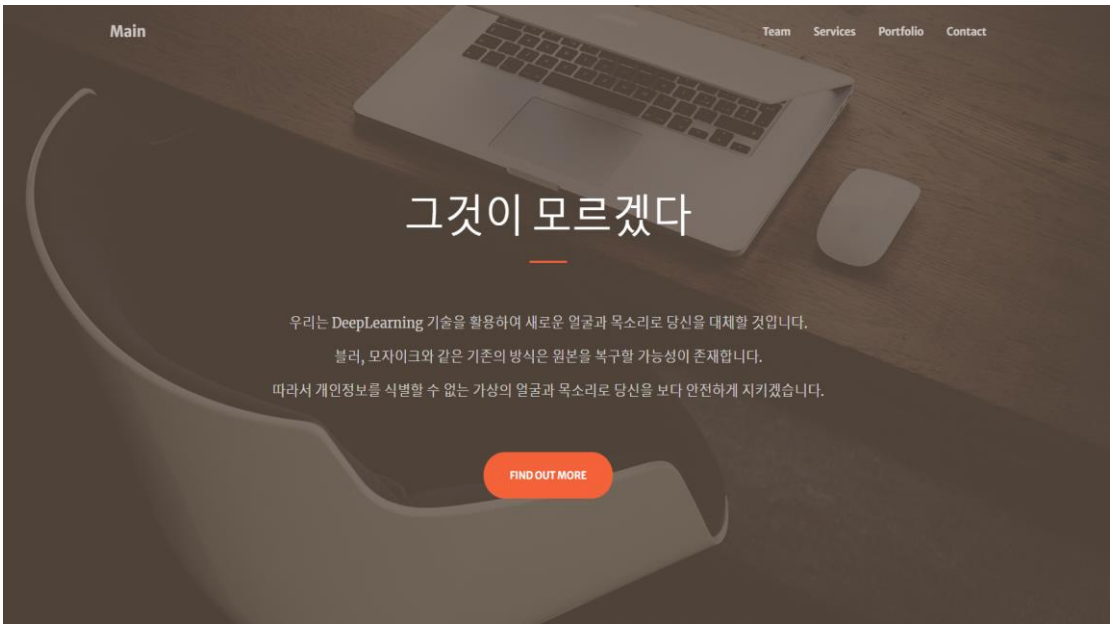
```
!nvcc --version
!ffmpeg -help
!pip3 install google-cloud-pubsub==1.7.0
!pip3 install google-cloud-storage==1.29.0
!pip3 install setuptools>=40.3.0
!pip3 install google-cloud-speech
```

```
!pip3 install google-cloud-translate==2.0.1
!pip3 install google-cloud-texttospeech
```

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
%cd /content
!git clone https://github.com/Rudrabha/Wav2Lip.git
!cp -
ri "/content/gdrive/MyDrive/Wav2lip/wav2lip_gan.pth" /content/Wav2
Lip/checkpoints/
!cd Wav2Lip && pip install -r requirements.txt
!wget "https://www.adrianbulat.com/downloads/python-fan/s3fd-
619a316812.pth" -O "Wav2Lip/face_detection/detection/sfd/s3fd.pth"
!cd Wav2Lip && python inference.py --
checkpoint_path checkpoints/wav2lip_gan.pth --
face "/content/gdrive/MyDrive/proj/proj_final1.mp4" --
audio "/content/gdrive/MyDrive/proj/proj_tts_clova1_1.wav" #0이 부분
을 수정함

output_file_name= "/content/Wav2Lip/results/result_voice.mp4" #res
ult_voice.mp4
from IPython.display import HTML
from base64 import b64encode

mp4 = open(output_file_name,'rb').read()
decoded_vid = "data:video/mp4;base64," + b64encode(mp4).decode()
HTML(f'<video width=800 controls><source src={decoded_vid} type="v
ideo/mp4"></video>')
```

6. 시연


Main
Team
Services
Portfolio
Contact
그것이 모르겠다
우리는 DeepLearning 기술을 활용하여 새로운 얼굴과 목소리로 당신을 대체할 것입니다.
블러, 모자이크와 같은 기존의 방식은 원본을 복구할 가능성이 존재합니다.
따라서 개인정보를 식별할 수 없는 가상의 얼굴과 목소리로 당신을 보다 안전하게 지키겠습니다.
FIND OUT MORE


포스코 Ai Big Data 15기 B1조
그것이 모르겠다
-15기, 대면 돌려도편-
포스코 Ai . Big Data Academy 15기 posco

7. 개선 기회 및 발전가능성

본 프로젝트에서는 안경, 마스크, 수염이 있는 경우 해당 얼굴을 탐지하고 swap하는 데 있어 제한이 있었다. 얼굴 위의 안경, 마스크, 수염 등을 탐지하여 swap하는 기술을 추가하는 방향을 개선할 수 있을 것이다.

또한, 음성의 경우 같은 문장이더라도 억양에 따라 의미가 다르기 때문에 text로 받고 바로 음성으로 구현하는 방식보다는 해당 음성의 억양을 비슷하게 구현하는 방식의 voice conversion을 구현해서 음성을 변조하는 방법을 향후에 활용하면 좋을 것이다.

추가적으로 다수의 인원이 화면 프레임에 잡히는 경우 특정 대상의 얼굴과 음성만을 바꾸는 기술을 구현하기 어려워서 이와 관련하여 개선할 필요가 있다.

딥페이크 기술들로 연예인과 일반인의 얼굴과 목소리가 악용될 여지가 이루어지고 있는데 본 프로젝트에서는 이러한 기술들이 개인정보 보호 측면에서 활용할 수 있다는 것을 보여주었다는 측면에서 발전 가능성이 있을 것이다. 다양한 방송 플랫폼의 증가로 인해 길가는 시민들의 얼굴이 화면에 잡히는 등 개인 초상권을 침해하는 문제에 해당 기술이 쓰일 수 있을 것이다. 추가적으로, 3D 모델링 기술을 추가하여 본인의 얼굴과 목소리를 활용한 가상인간을 생성할 수 있다고 기대할 수 있다.

[그림] 가상 인간 루이

참고문헌(Appendix)

권철홍 (2021). 딥러닝 기반 한국어 실시간 TTS 기술 비교. The Journal of the Convergence on Culture Technology (JCCT), 7(1), 640-645.

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., & Aila, T. (2020). Training generative adversarial networks with limited data. arXiv preprint arXiv:2006.06676.

Kim, J., Kim, S., Kong, J., & Yoon, S. (2020). Glow-TTS: A generative flow for text-to-speech via monotonic alignment search. arXiv preprint arXiv:2005.11129.

Kong, J., Kim, J., & Bae, J. (2020). Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. arXiv preprint arXiv:2010.05646.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., ... & Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4681-4690).

Li, Y., & Lyu, S. (2018). Exposing deepfake videos by detecting face warping artifacts. arXiv preprint arXiv:1811.00656.

Nguyen, T. T., Nguyen, C. M., Nguyen, D. T., Nguyen, D. T., & Nahavandi, S. (2019). Deep learning for deepfakes creation and detection: A survey. arXiv preprint arXiv:1909.11573.

Prajwal, K. R., Mukhopadhyay, R., Namboodiri, V. P., & Jawahar, C. V. (2020). A lip sync expert is all you need for speech to lip generation in the wild. In Proceedings of the 28th ACM International Conference on Multimedia (pp. 484-492).

Siarohin, A., Lathuilière, S., Tulyakov, S., Ricci, E., & Sebe, N. (2019). First order motion model for image animation. Advances in Neural Information Processing Systems, 32, 7137-7147.

Sisman, B., Yamagishi, J., King, S., & Li, H. (2020). An overview of voice conversion and its challenges: From statistical modeling to deep learning. IEEE/ACM Transactions on Audio, Speech, and Language Processing.

8. Learned Lessons

- **박규석 :** 먼저 이번 과정을 진행하면서 비대면이라는 큰 적을 만나 과정의 진행이 어려웠으나 너무나 좋은 조원과 교수님들을 만나 이 어려운 과정들 속에서도 뭐 하나 놓치지 않고 배울 수 있게 되어 너무 너무나 감사합니다. 특히 ai는 내가 할 수 없는 것으로 생각했지만, 이 과정을 통해서 ai를 알아가고 ai를 활용하며 제 생각을 변화시키며 저를 성장시키는 과정이었습니다. 다시 한 번으로 B1조 분들을 만나게 되어서 정말 좋은 시간이었습니다. 감사합니다. 아잉~ B1조 사랑행~~~~~

- **채다미 :** 이번 과정을 통해 AI를 마스터한 게 아닌 이제 시작입니다! (공부할 건 정말 많다는 뜻^^!) 다만 공부하고자 하는 AI 분야의 지식을 접하고, 찾아보고, 활용해보는 법에 대해서 익힌 과정들이었습니다. B1조랑 같은 조를 하게 해 주셔서 감사합니다! 곁에 있는 조원들 한 명 한 명 배울 점이 가득한 멋진 사람들이었습니다. 무언가라도 얻은 게 하나라도 있다면 그 과정 자체가 의미 있는 게 아닐까요? 그렇다면 조원들을 얻었으니 AI 과정도 제게 의미 있는 시간들이었습니다. B1 사당행~ (사랑은 안될까~?)

- **임태미 :** AI의 '에'도 모르는 문송을 이렇게 잘 이끌어준 팀원들에게 너무 너무 너무 감사하다는 말씀 전합니다..<3 선택과 집중으로 프로젝트에 누가 되지 않으려고 노력했는데 저와 같은 생각이라면 정말 다행입니다..<3 여튼 쨌든 ♥B1 탬담먕귯누정썬 사당행♥

- **서정빈 :** 2개월이 넘도록 빡빡한 스케쥴과 수많은 할 일들 속에서 많은 고난이 존재했지만, 그럼에도 이 과정을 끝까지 버틸 수 있었던 건 모두 팀원들 덕분입니다. 해당 과정 끝날 때까지 다툼도 없이 완전체로 계속 유지되서 더 소중한 인연이라고 생각합니다. 혼자 했었다면 이런 수준의 결과물을 낼 수 있었나 싶습니다. 같이 밤샘하고 고통받으면서도 (미쳐서) 웃고 떠들고 한 날들이 이제는 추억이 되었습니다. 다시 한 번 정말 사ㄹ....아니 사당합니다.... ^3^

- **김미향 :** 7월에 시작했지만 코로나 여파로 온라인 수업으로 8월부터 시작하여 10월에 마무리하는 여정 중에 배우는 내내 힘들기도 했지만 이렇게 하루를 온전히 통으로 공부하는데 쓸 수 있어서 행복하고 기뻤습니다. 내가 원하는 것을 만드는 것에서 벗어나 함께 무언가를 만들어 가는 걸 배워볼 수 있는 경험이 되었습니다. 늘 긍정적이고 순둥순둥하고 빛과 소금의 능력자들이 모인 B1조 사당으로 집합이 아니라...사당해입니다ㅎㅎ

- **임영선 :** 짧은 시간이었지만 과정을 수행해나가면서 많은 걸 배운 것 같습니다. 처음하는 ai 프로젝트라 걱정이 많았는데 좋은 팀원들을 만나 즐겁게 무사히(?) 결과물을 낼 수 있었습니다. 수업과 과제 프로젝트까지 힘든 고생길이었지만 새벽까지 과제방 운영하며 오손도손 잘 해낸 B1에게 이 영광 돌립니다. B1 사당행!

B반 1조 Faceswap과 GAN을 이용한 신변보호 서비스

- **박진우 :** AI 과정을 통해 많은 지식과 경험을 얻을 수 있었지만 이걸 보시는 16기 분들에게 정말 해드리고 싶은 말은 무엇보다 협업을 통해서 얻어가는 게 정말 많습니다!! 의지할 수 있는 훌륭한 조원들과 함께해서 과정의 진행에 있어서 부족한 부분을 메울 수 있었고 이를 통해 크게 발전할 수 있었습니다!!!!! 실력적인 성장만큼 소중한 인연들을 만나서 행복합니다 ㅎㅎㅎㅎㅎ 3달간의 과정 속에서 함께 고생해서 너무 고맙고 B1조 사랑해요~~~~!!