Re:Mind

4분반 5조
김다혜, 선다혜, 오찬희, 심우석

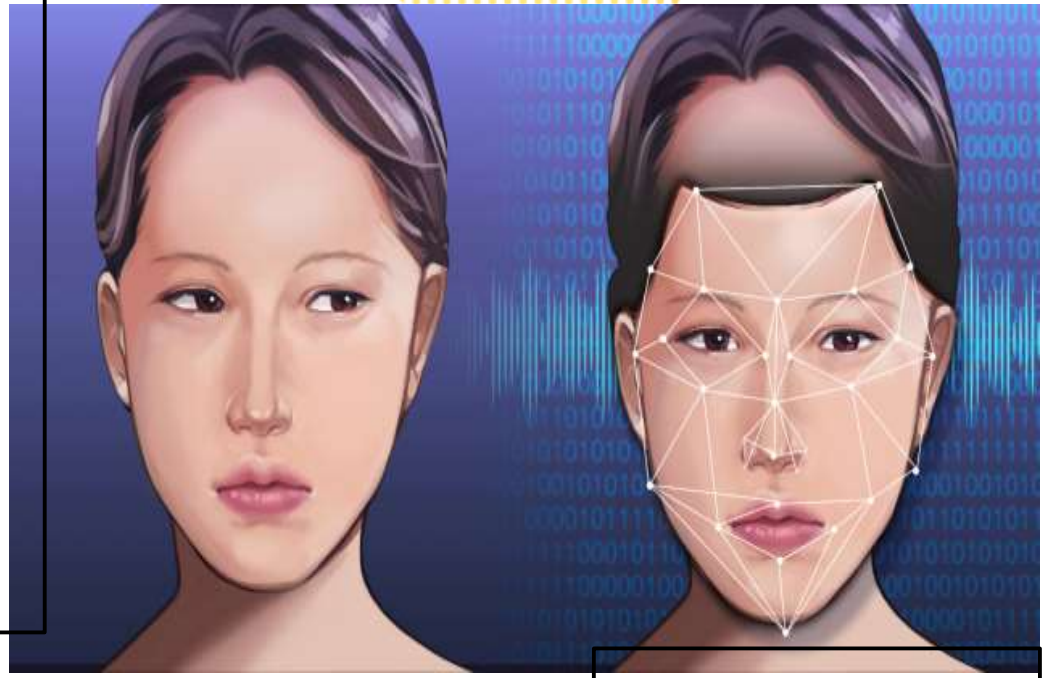


TABLE OF CONTENTS

01

Motivation & Objective

Project (Re:Mind)
Motivation & Objective

Motivation (Reminiscence)

너를 만났다: '가상현실' 속 그리운 사람과의 재회, 실제 치유가 될까?

김홍빈
BBC 코리아

2020년 2월 14일

故 터틀맨 AI 기술로 복원…12년 만에 '거북이' 완전체 무대

김지원 ■ / 기사승인 : 2020-12-10 09:50:55

f ✓ 𝕤 ℗ N b

Mnet '다시 한번'…그리운 아티스트 음성·모습 복원 AI 음악 프로젝트

혼성 그룹 거북이가 12년 만에 완전체로 무대에 섰다. Mnet 'AI 프로젝트 다시 한번'(이하 '다시 한번')을 통해 AI 기술로 복원된 터틀맨과 함께했다.

"안중근 의사, 인공지능으로 되살아나다"

X 윤영주 기자 ⊙ 입력 2021.06.28 19:07 ▤ 댓글 2 ♡ 좋아요 0

'대한황실문화의 관리·지원과 디지털 복원 방안' 주제 정책포럼
철저한 고증 기반, AI 기술로 안중근 의사 모습을 생생하게 재현
㈜비빔블, 관객과 실시간 상호 소통 가능한 AI 디지털 휴먼 제작
국내 AI 디지털 휴먼 기술 활용한 역사인물 복원 가능성 열어

Motivation (Fun)

Motivation (Compare other Apps)

- Not a smartphone app
- Paid Service

- Just FaceSwap (Deepfake)
- Vulnerable to security

- Just Lip-Syncing
- Bad User Experience

Objective

DeepFake (Face Swap)

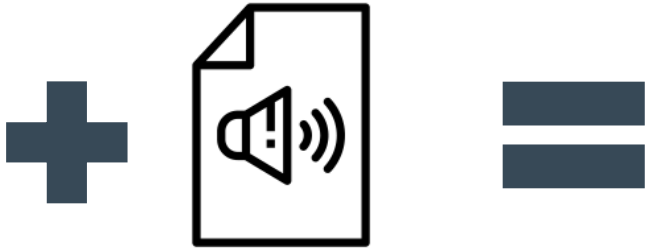
Lip-Syncing (Using My voice file)

Objective

DeepFake (Face Swap) + Lip Syncing (+My Voice)

Source Picture

Source Voice File

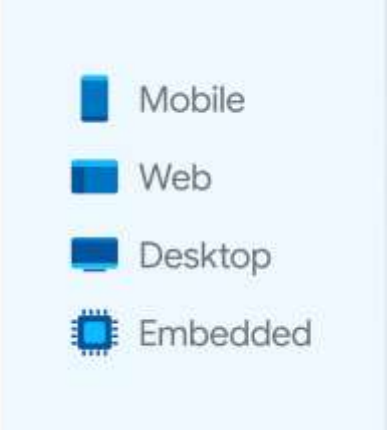


Result MP4 file

Objective

Cross Platform Application (Web + Mobile)

Flutter

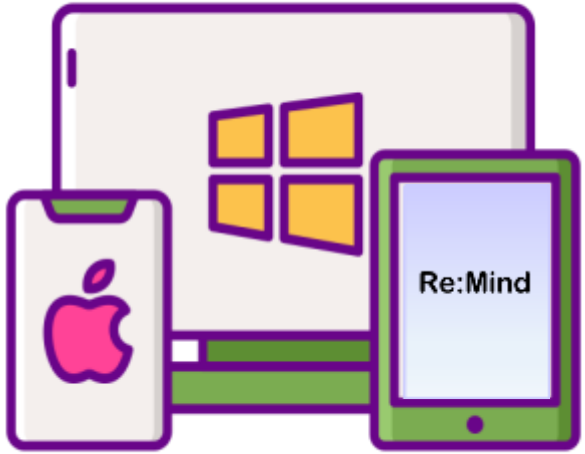


02

Project Overview

Project Overview,
Structure, Environment

Project Overview



Key Features

- For fun & reminiscence
- DeepFake (FaceSwap) + Lip Syncing
- Cross Platform App (Windows, Mac, Android)
  - Easy To Use
  - Better UI/UX
- Training One-shot (Picture)
- High Quality + Done Quickly
- Various templates

Project Overview

Divide into two tracks

1. Gif file (3~4 seconds) + No Voice file

Source Picture

=

Result Gif file

Project Overview

Divide into two tracks

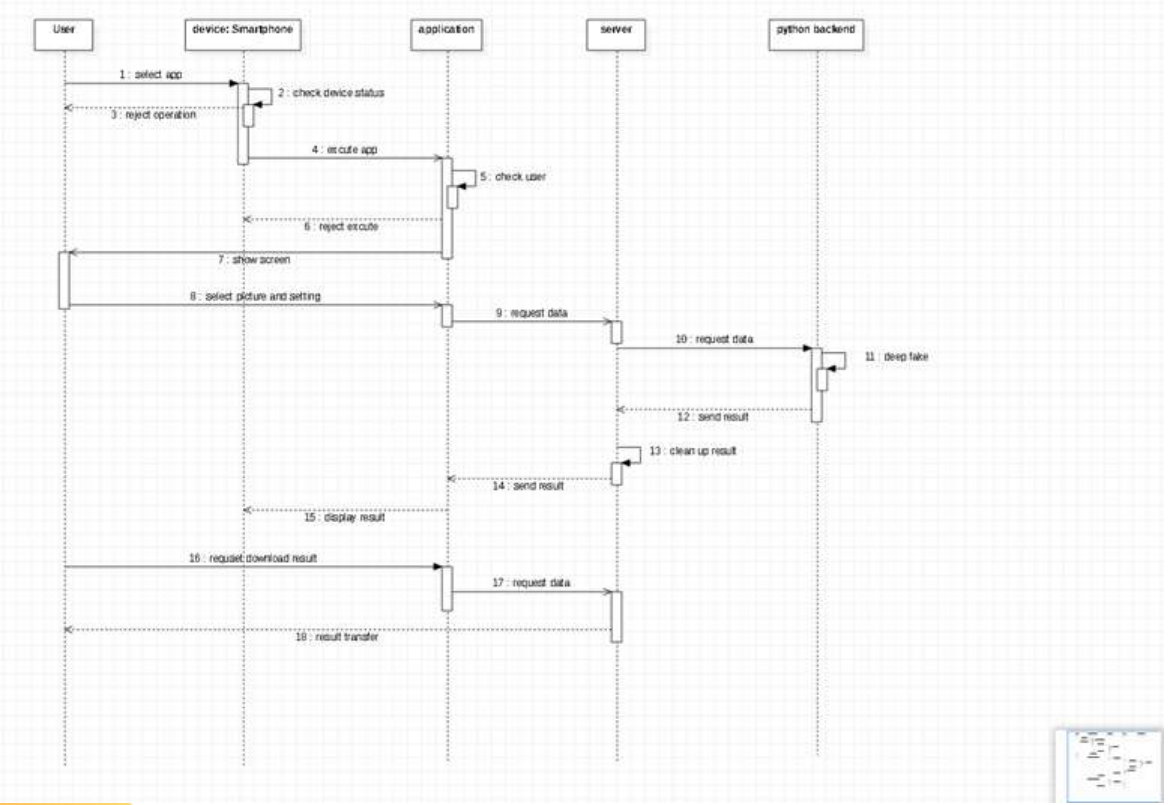
2. MP4 file (30~60 Seconds) + Voice file

Source Picture

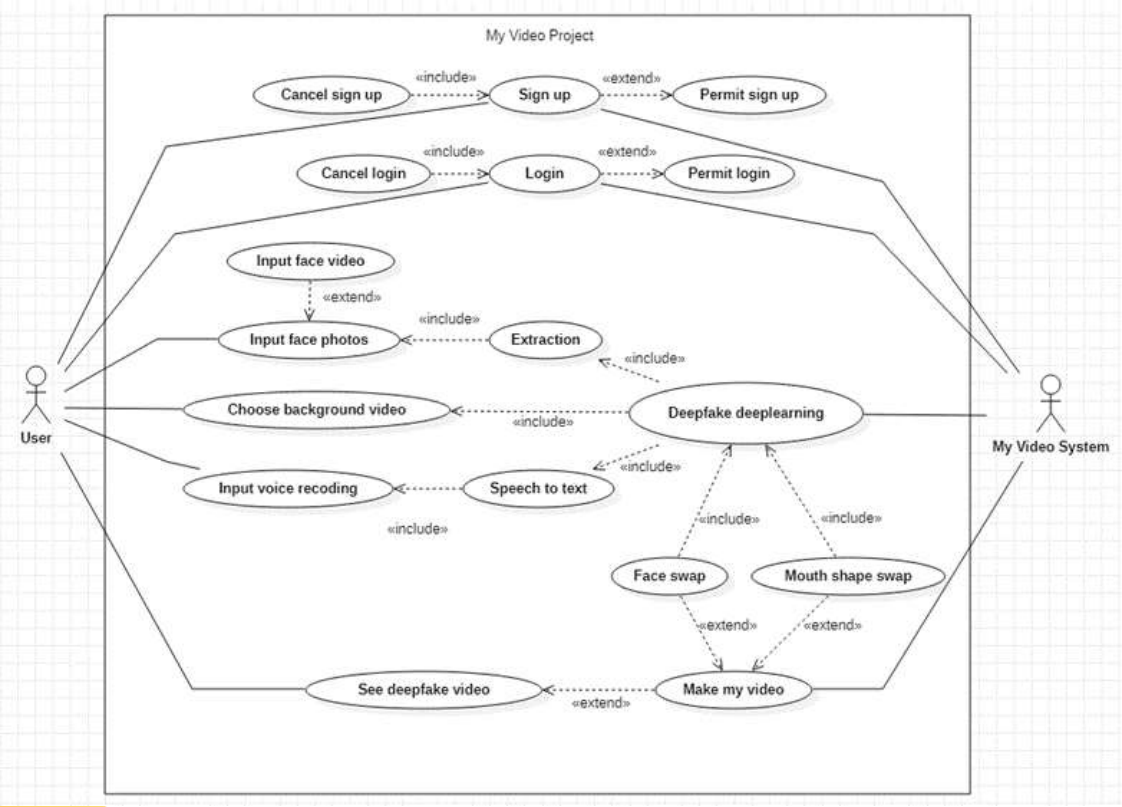
Source Voice File

Result MP4 file

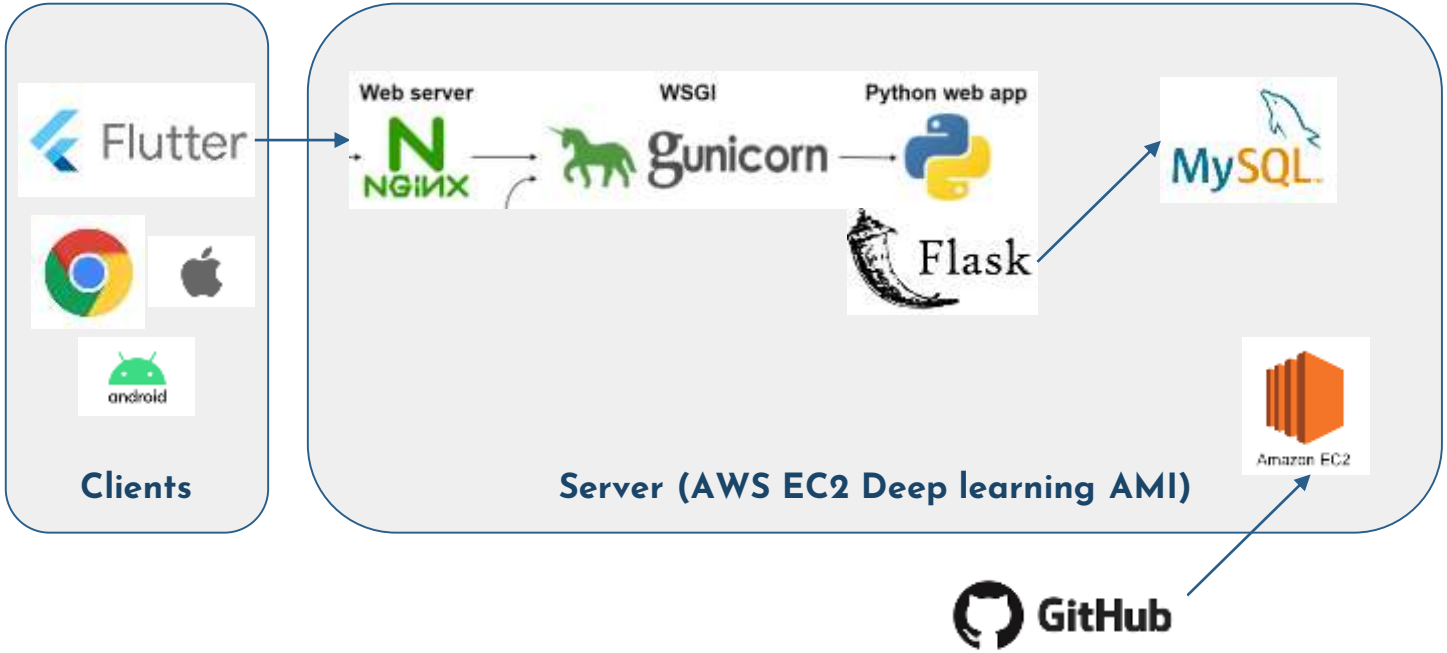
Sequence Diagram



Usecase Diagram



Structure


Flutter
android
Clients
Web server
NGINX
WSGI
gunicorn
Python web app
Flask
MySQL
Amazon EC2
Server (AWS EC2 Deep learning AMI)
GitHub

03

Technology Used

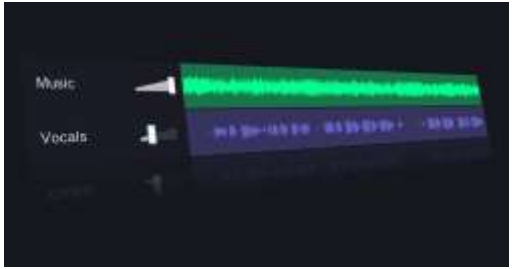
Deepfake (SimSwap),
LipSyncing (Wav2lip),
Spleeter, Flutter

Model

DeepFake
+

+
Voice Extraction

Lip-Syncing



※ Requirements - Training Time, Convenience

Model - Deep Fake (Ref.SimSwap)

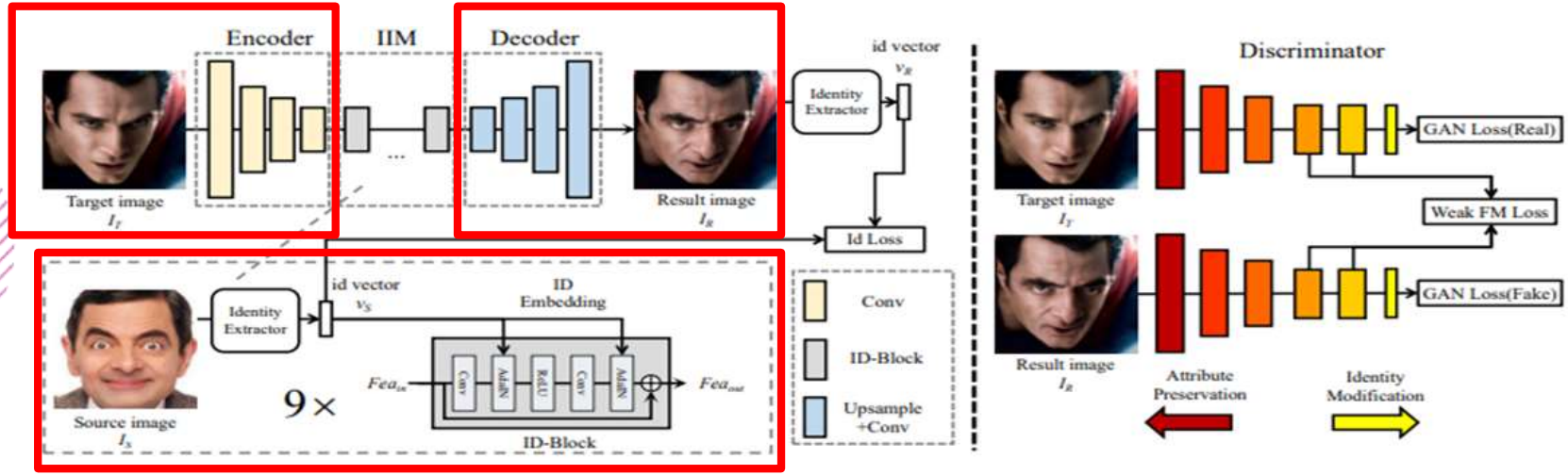
~~FUNIT + SPADE + AdaIN~~

↓

PatchGAN + Pix2PixHD + AdaIN

Model - Deep Fake (Ref.SimSwap)

Overcome the defects in generalization and attribute preservation



Generalization to Arbitrary Identify + Preserving the Attributes of the Target

Model - Deep Fake (Ref.SimSwap)

Image To Gif

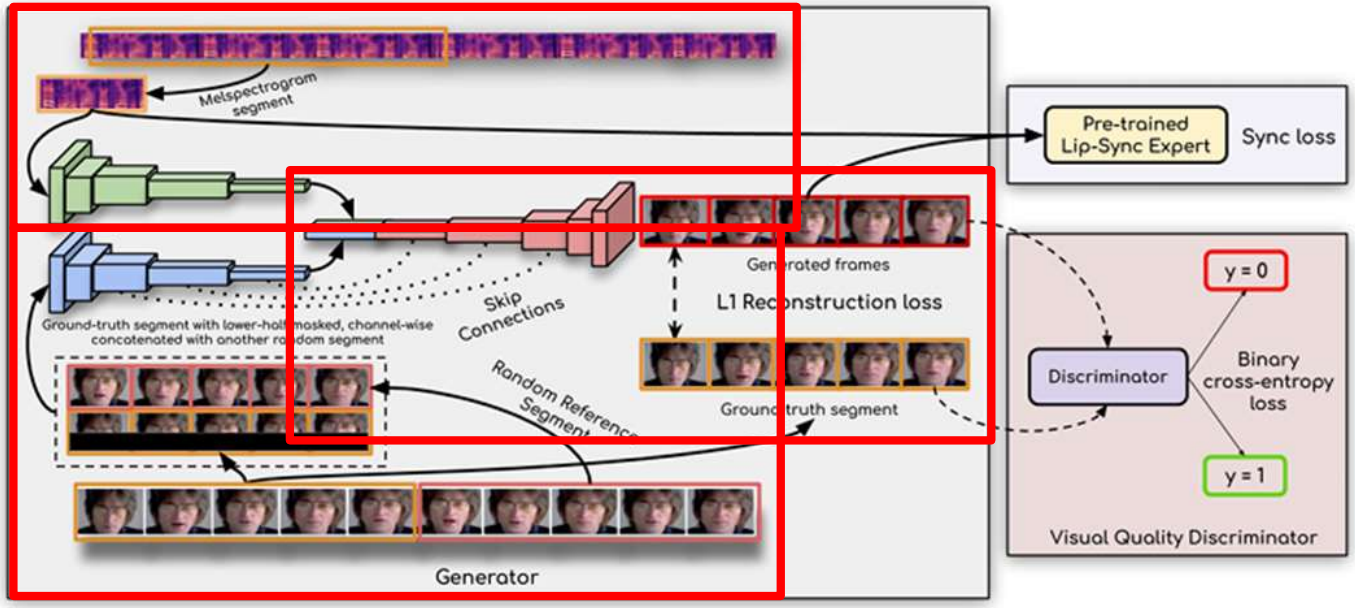
Source

Target

Result

Model - Lip Syncing (Ref.Wav2lip)

Extraction -> Training -> Converting



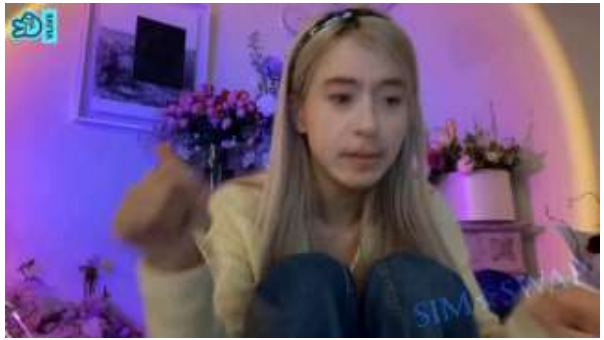
Model - Lip Syncing (Ref.Wav2lip)

Image + Voice To MP4

Source

Target

Result



Model - Voice Extraction (Ref.Spleeter)

pytest passing | python 3.6 | 3.7 | 3.8 | 3.9 | pypi package 2.3.0 | deezer-research v2.2.1 | docker pulls 21k | Open in Colab

chat on gitter | JOSS 10.21105/joss.02154

⚠ Spleeter 2.1.0 release introduces some breaking changes, including new CLI option naming for input, and the drop of dedicated GPU package. Please read CHANGELOG for more details.

Extract only voice from mp3, mp4 or wav file

→ Better Quality !!

Cross Platform Application (Flask + Flutter)

Python Application & Cross Platform

04

Implementation Detail

Implementation Detail,

Model - Python Library

- **Deepfake (SimSwap) - Insightface, torch, torchVision, Cuda, cv2, tensorflow**
- **Lip-Syncing (Wav2lip) - Opencv, torch, torchVision, Cuda, librosa, tensorflow**
- **Voice Extraction (Spleeter) - tensorflow, ffmpeg-python, norbert, librosa, typer**

Model - Deepfake (Ref.SimSwap)

1. Set Image Size

```
transformer = transforms.Compose([
        transforms.ToTensor(),
        #transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])

transformer_Arcface = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])

detransformer = transforms.Compose([
        transforms.Normalize([0, 0, 0], [1/0.229, 1/0.224, 1/0.225]),
        transforms.Normalize([-0.485, -0.456, -0.406], [1, 1, 1])
    ])
```

2. Set Option (Weights)

```
# %%
opt = TestOptions()
opt.initialize()
opt.parser.add_argument('-f') ## dummy arg to avoid bug
opt = opt.parse()
opt.pic_a_path = './demo_file/Iron_man.jpg' ## or replace it with image from your own google drive
opt.video_path = './demo_file/multi_people_1080p.mp4' ## or replace it with video from your own google drive
opt.output_path = './output/demo.mp4'
opt.temp_path = './tmp'
opt.Arc_path = './arcface_model/arcface_checkpoint.tar'
opt.isTrain = False
opt.use_mask = True  ## new feature up-to-date
```

Model - Deepfake (Ref.SimSwap)

3. Convert (Main)

```
def convert(image):
    with torch.no_grad():
        #pic_a = opt.pic_a_path
        # img_a = Image.open(pic_a).convert('RGB')
        #img_a_whole = cv2.imread(image)

        imageStr = base64.b64decode(image)
        nparr = np.fromstring(imageStr, np.uint8)
        img_a_whole = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
        img_a_align_crop, _ = app.get(img_a_whole,crop_size)
        img_a_align_crop_pil = Image.fromarray(cv2.cvtColor(img_a_align_crop[0],cv2.COLOR_BGR2RGB))
        img_a = transformer_Arcface(img_a_align_crop_pil)
        img_id = img_a.view(-1, img_a.shape[0], img_a.shape[1], img_a.shape[2])

        # convert numpy to tensor
        img_id = img_id.cuda()
```

Model - Lip-Syncing (Ref.Wav2lip)

1. Data Parse

```
parser = argparse.ArgumentParser(description='Inference code to lip-sync videos in the wild using Wav2Lip models')

parser.add_argument('--checkpoint_path', type=str,
                    help='Name of saved checkpoint to load weights from', required=False)
parser.add_argument('--face', type=str,
                    help='Filepath of video/image that contains faces to use',
                    default='D:/SimSwap/SimSwap/output/demo.mp4', required=False)
parser.add_argument('--audio', type=str,
                    help='Filepath of video/audio file to use as raw audio source',
                    default='D:/SimSwap/SimSwap/output/demo.wav', required=False)
parser.add_argument('--outfile', type=str, help='Video path to save result. See default for an e.g.',
                            default='D:/SimSwap/SimSwap/output/result_voice.mp4')

parser.add_argument('--static', type=bool,
                    help='If True, then use only first video frame for inference', default=False)
parser.add_argument('--fps', type=float, help='Can be specified only if input is a static image (default: 25)',
                    default=25., required=False)

parser.add_argument('--pads', nargs='+', type=int, default=[0, 20, 0, 0],
                    help='Padding (top, bottom, left, right). Please adjust to include chin at least')
```

2. Face Detect

```
def face_detect(images):
    detector = face_detection.FaceAlignment(face_detection.LandmarksType._2D,
                                            flip_input=False, device=device)

    batch_size = args.face_det_batch_size

    while 1:
        predictions = []
        try:
            for i in tqdm(range(0, len(images), batch_size)):
                predictions.extend(detector.get_detections_for_batch(np.array(images[i:i + batch_size])))
        except RuntimeError:
            if batch_size == 1:
                raise RuntimeError('Image too big to run face detection on GPU. Please use the --resize_f
            batch_size //= 2
            print('Recovering from OOM error; New batch size: {}'.format(batch_size))
            continue
        break
```

Model - Lip-Syncing (Ref.Wav2lip)

3. Data Generation

```python
def datagen(frames, mels):
    img_batch, mel_batch, frame_batch, coords_batch = [], [], [], []

    if args.box[0] == -1:
        if not args.static:
            face_det_results = face_detect(frames)# BGR2RGB for CNN face detection
        else:
            face_det_results = face_detect([frames[0]])
    else:
        print('Using the specified bounding box instead of face detection...')
        y1, y2, x1, x2 = args.box
        face_det_results = [[f[y1: y2, x1:x2], (y1, y2, x1, x2)] for f in frames]

    for i, m in enumerate(mels):
        idx = 0 if args.static else i%len(frames)
        frame_to_save = frames[idx].copy()
        face, coords = face_det_results[idx].copy()

        face = cv2.resize(face, (args.img_size, args.img_size))
```

4. Load Model & Weights

```python
def _load(checkpoint_path):
    if device == 'cuda':
        checkpoint = torch.load(checkpoint_path)
    else:
        checkpoint = torch.load(checkpoint_path,
                                map_location=lambda storage, loc: storage)
    return checkpoint

def load_model(path):
    model = Wav2Lip()
    print("Load checkpoint from: {}".format(path))
    checkpoint = _load(path)
    s = checkpoint["state_dict"]
    new_s = {}
    for k, v in s.items():
        new_s[k.replace('module.', '')] = v
    model.load_state_dict(new_s)

    model = model.to(device)
    return model.eval()
```

Model - Lip-Syncing (Ref.Wav2lip)

5. Main

```
def main():
    if not os.path.isfile(args.face):
        raise ValueError('--face argument must be a valid path to video/image file')

    elif args.face.split('.')[1] in ['jpg', 'png', 'jpeg']:
        full_frames = [cv2.imread(args.face)]
        fps = args.fps

    else:
        video_stream = cv2.VideoCapture(args.face)
        fps = video_stream.get(cv2.CAP_PROP_FPS)

        print('Reading video frames...')

        full_frames = []
        while 1:
            still_reading, frame = video_stream.read()
            if not still_reading:
                video_stream.release()
                break
            if args.resize_factor > 1:
                frame = cv2.resize(frame, (frame.shape[1]//args.resize_factor, frame
```

Model - Voice Extraction (Ref.Spleeter)

Main (In Lip-Syncing)

```
video_path = "D:/SimSwap/SimSwap/output/demo.mp4"
audio_path = "D:/SimSwap/SimSwap/output/demoa.wav"

spleeter_wav_path = "D:/SimSwap/SimSwap/output/after.wav"


def get_stems(filePath, fileSavePath):
    separator = Separator('spleeter:2stems')
    separator.separate_to_file(filePath, fileSavePath, codec="wav", bitrate="128k")


def set_path(vid, aud):
    video_path = vid
    audio_path = aud

get_stems(audio_path, spleeter_wav_path)
inference.convert(video_path, spleeter_wav_path)
```

Flask (Backend)

1. Gif transformation

```python
@app.route('/face_swap', methods = ['GET', 'POST'])
def face_swap():
    data = request.get_json()

    if 'image' not in data:
        return "", 400
    elif 'templete' not in data:
        return "", 400
    main.convert(data['image'], data['templete'])
```

2. MP4 transformation

```python
#임시 링크
video_path = 'D:/SimSwap/SimSwap/output/demo.mp4'
audio_path = 'D:/SimSwap/SimSwap/output/demoa.wav'

@app.route('/lip', methods = ['GET', 'POST'])
def lip():
    Wav2Lip.main.set_path(video_path, audio_path)
    Wav2Lip.main()
```

Flutter (Frontend)

Modeling -> Return the Result

```
try {
  var encodedData = await compute(base64Encode, imageData);
  Response response = await dio.post('http://10.0.2.2:5000/face_swap',
      data: {
        'image': encodedData,
        'templete': templeteData
      }
  );
  String result = response.data;
  return compute(base64Decode, result);
} catch (e) {
  return null;
}
```

1st Mentoring Feedback

1. Processing speed, model capacity problem
   → **Lightweight model**

1. Wav2lip model is a model tailored to English
   → Since the **expected user is Korean**,
   training after **changing the data set**.

Lightweight model

To shorten the learning time,
The Training file is built in advance.

wav2lip.pth	2021-12-04 오후 4:57	PTH 파일	425,594KB
wav2lip_gan.pth	2021-12-07 오전 1:07	PTH 파일	425,588KB

arcface_checkpoint	2021-11-30 오전 4:51	ALZip TAR File	748,898KB

latest_net_D1.pth	2020-04-22 오후 9:05	PTH 파일	27,213KB
latest_net_D2.pth	2020-04-22 오후 9:05	PTH 파일	27,213KB
latest_net_G.pth	2020-04-22 오후 9:06	PTH 파일	215,082KB

Lightweight - Deepfake

Parameter was **changed** in consideration of Running Time & Quality

```
hparams = HParams(
    num_mels=70,  # Number of mel
    #  network
    rescale=True,  # Whether to r
    rescaling_max=0.7,  # Rescali

    # Use LWS (https://github.com
    # It"s preferred to set True
    # Does not work if n_ffit is
    use_lws=False,

    n_fft=800,  # Extra window si
    hop_size=200,  # For 16000Hz,
    win_size=800,  # For 16000Hz,
    sample_rate=16000,  # 16000Hz
```

num_mels, rescaling_max, batch_size, num_workers, checkpoint_interval, eval_interval, syncnet_batch_size, syncnet_eval_interval, Syncnet_checkpoint_interval etc…

Lightweight - Lip-syncing

Parameter was **changed** in consideration of
Eval model's sync & train loss

```
###################### Our training parameters ################################
img_size=96,
fps=25,

batch_size=12,
initial_learning_rate=1e-4,
nepochs=200000000000000000000,  ### ctrl + c, stop whenever eval loss is consistentl
num_workers=12,
checkpoint_interval=2000,
eval_interval=2000,
save_optimizer_state=True,

syncnet_wt=0.0, # is initially zero, will be set automatically to 0.03 later. Lea
syncnet_batch_size=32,
syncnet_lr=1e-4,
syncnet_eval_interval=5000,
syncnet_checkpoint_interval=5000,
```

Mel_step_size, learning_rate, Fps (frame), nepochs, num_workers, syncnet_wt, syncnet_batch_size, syncnet_eval_interval etc…

Changing weight dataset - Lip-syncing

Compare them according to the environment of each dataset(LRW, LRS2, LRS3)

Translating sounds from the Eastern languages: LRS3 > LRS2

But LRS3 has a lot of artifacts around the face.

Use LRS2

Lightweight result

Computing Specs

Processor - Intel i5-6600

RAM - 16GB

Graphics - GTX 960

Face Swap

- Ex) 3 ~ 4 Seconds GIF
  20 Seconds → **14 Seconds**

- Ex) 30 Seconds Video
  3 Min → **2 Min**

Lip sync

- Ex) 30 Seconds Video
  2 Min → **1.5 Min**

Voice extraction

- Ex) 30 Seconds Voice
  10 Seconds → **10 Seconds**

2nd Mentoring Feedback

3. Cloud Server Instance & Cost Problems

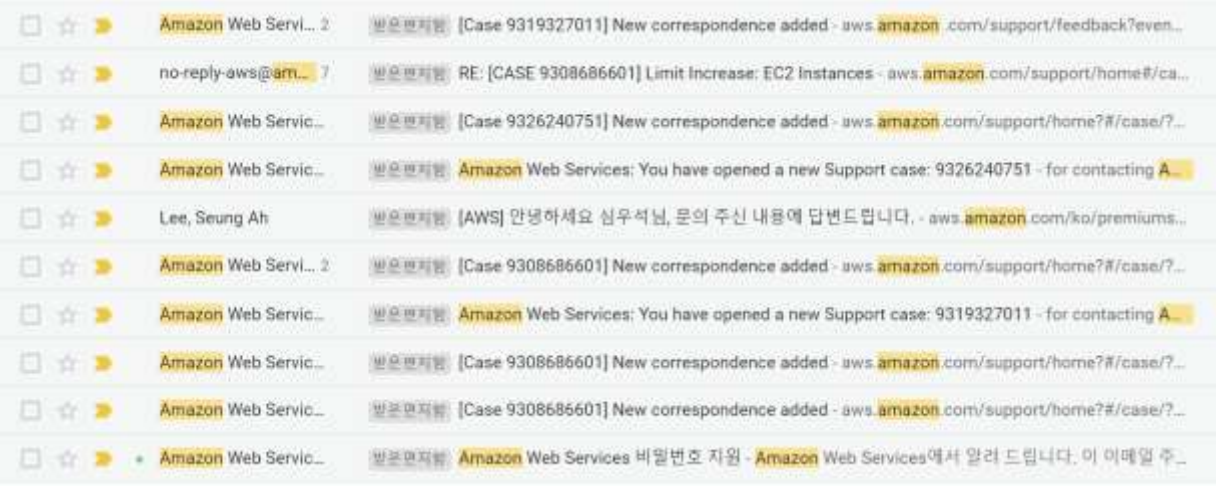
→ We must use CUDA, Need GPU Server
So, **Looking for a solution**

Cloud Server Instance & Cost

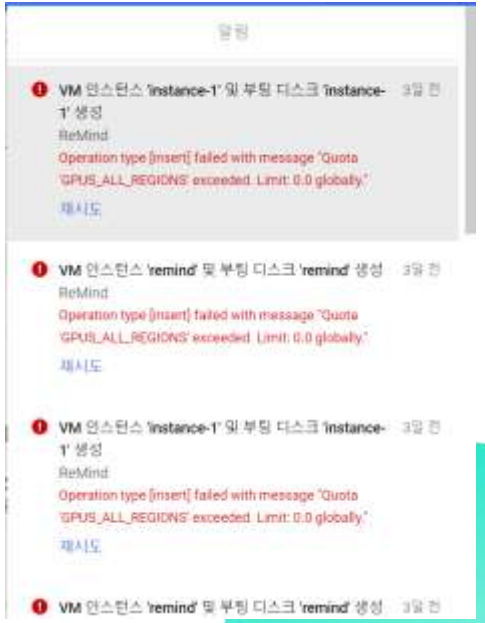
We are using the CUDA -> Must use GPU Server

- Trying to deploy the Cloud GPU Server

AWS Cloud Service



Google Cloud Service



Cloud Server Instance & Cost

Very Expensive Server cost → **Looking for a solution**

AWS Cloud Service

P2 인스턴스 세부 정보

이름	GPU	vCPU	RAM(GiB)	네트워크 대역폭	시간당 요금*	시간당 RI 요금**
p2.xlarge	1	4	61	높음	0.900 USD	0.425 USD

Google Cloud Service

월별 예상 가격

US$255.22

시간당 약 US$0.35

You have US$358,067.00 free trial credits remaining

사용한 만큼만 비용 지불: 선불 비용 없이 초당 청구

Naver Cloud Service

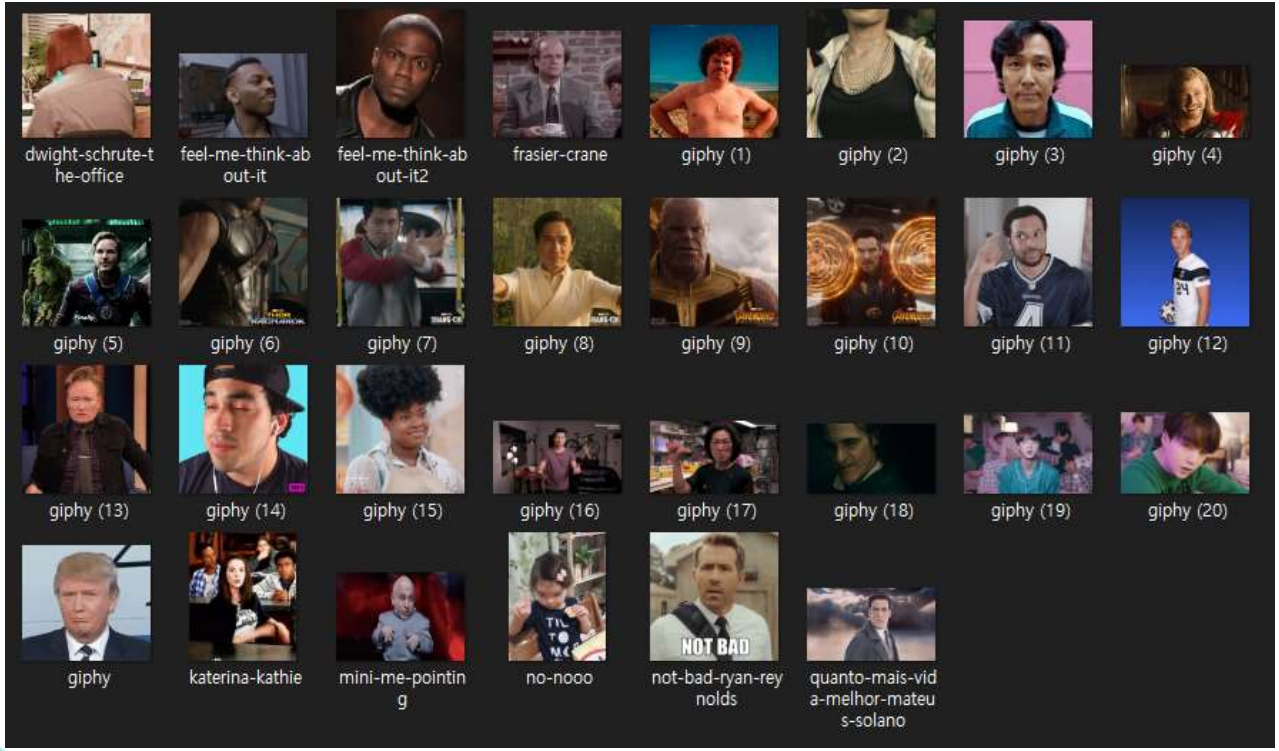
제공 사양					이용 요금	
GPU	GPU 메모리	vCPU	메모리	디스크	시간	월
1개	24GB	4개	30GB	50GB(HDD) / 100GB(HDD)	1,667원 / 1,671원	1,200,000원 / 1,202,880원
				50GB(SSD) / 100GB(SSD)	1,671원 / 1,679원	1,202,880원 / 1,208,640원

05

Progress

Application demo

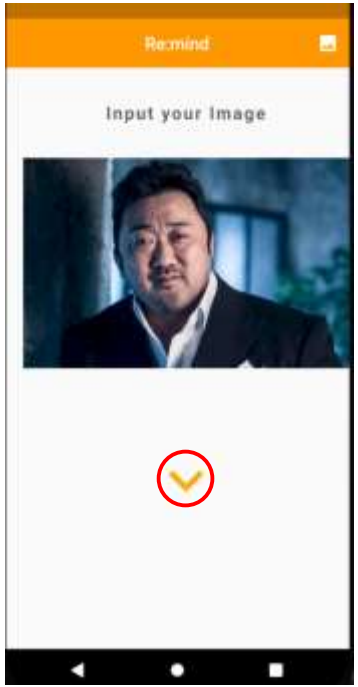
Template (GIF)

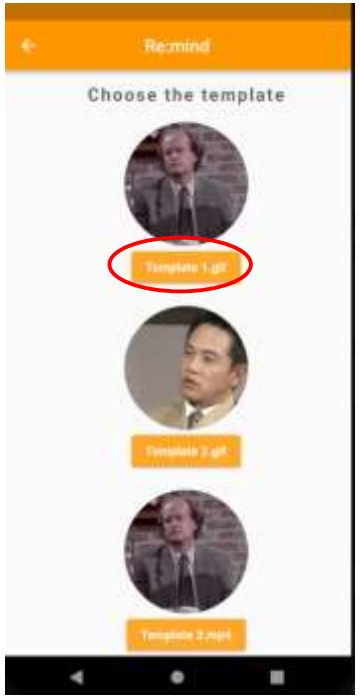


Template (MP4)

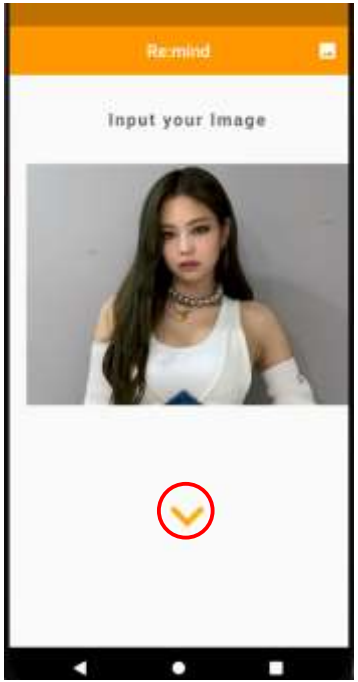
Flutter Demo

Flutter Demo

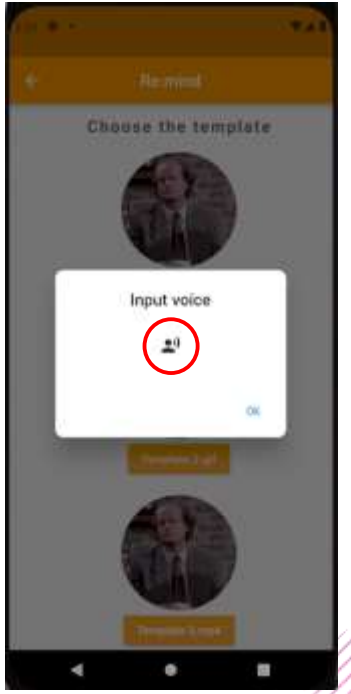




Flutter Demo



Flutter Demo



GIF Template Demo

Source Picture

Target GIF

=

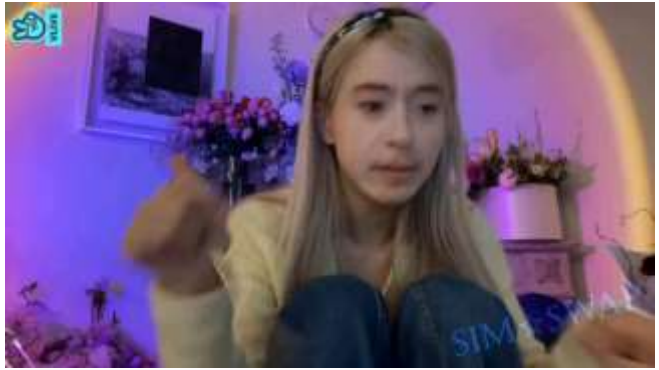
Result Gif file

MP4 (Video) Template Demo

Source Picture

Source Voice File

Result MP4 file



06

Role & Plan

Members Role
& Plan

OUR TEAM Role

심우석 201636417

qkqh8639@gmail.com

- Lightweight, Server

오찬희 201735855

fasvvc@gmail.com

-Modeling, Lightweight

선다혜 201835466

adad05011@gachon.ac.kr

- Modeling, flutter

김다혜 201835414

ekgp3987@naver.com

- Modeling, flutter

Plan

01

2021 - 3 ~ 8

Project Planning,
Related data collection

02

2021 - 9 ~ 12

App Implementation,
Model Lightweight

03

2022 - 1 ~ 2

Flutter UI Update,
Server Setting

04

2022 - 3 ~ 6

Service deployment,
Documentation

Wiki & YouTube

GitHub https://github.com/dntjr41/Graduation_project/wiki

YouTube Youtube URL link

RESOURCES

Papers & Web Sites

- https://giphy.com/
- https://arxiv.org/pdf/2008.10010.pdf
- https://arxiv.org/pdf/2106.06340v1.pdf
- https://anaconda.org/deezer-research/spleeter

Github Pages

- https://github.com/neuralchen/SimSwap
- https://github.com/deepinsight/insightface
- https://github.com/zllrunning/face-parsing.PyTorch
- https://github.com/deezer/spleeter
- https://github.com/Rudrabha/Wav2Lip
- https://github.com/r9y9/deepvoice3_pytorch
- https://github.com/1adrianb/face-alignment

THANKS

Do you have any questions?

4분반 5조 -김다혜, 선다혜, 오찬희, 심우석

담당 교수님 - 한명묵 교수님

가천대학교
Gachon University