

# 음성 및 영상합성을 통한 Fake Video 생성

Deep Learning Application based on StarGAN-VC and FSGAN  
2조: Advance

## Team Members

2조 - 김정준, 김용신, 김예현, 박승우, 장우원

## Schedule

2020.10.02 데이터수집 및 주제선정  
2020.10.09 데이터 전처리, 논문리뷰  
2020.10.16 모델 생성 및 훈련, 결과도출  
2020.10.23 성능향상을 위한 튜닝  
2020.10.30 문서작성 및 프로젝트 발표

## Preparation

Data Set : 유명인 음성Data  
GAN(Generative Adversarial Network) 기반 기술  
StarGANVC / FSGAN 모델 적용

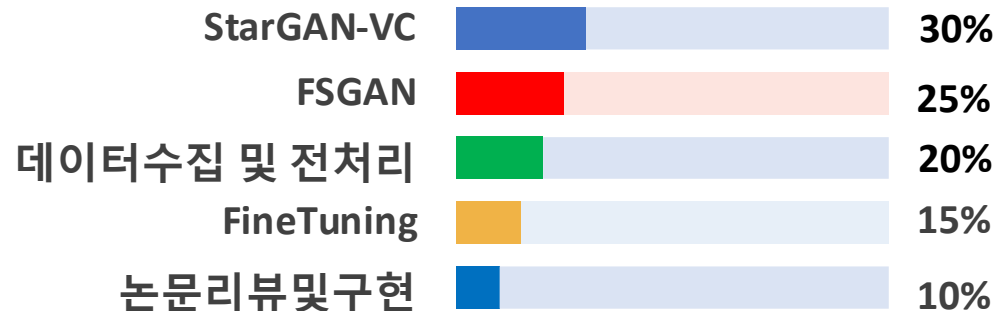
## GAN을 활용한 음성&영상 합성



Face Swap과 Voice Conversion에 관한 연구

- FSGAN기반 Face Swap
- StarGAN-VC를 이용한 음성 변조

## Works



# Motivation

## DEEPBRAIN 의 AI Human 기술 간략하게 구현해보기 StarGAN을 이용한 음성 변환과 FSGAN을 이용한 영상 합성을 통해 AI Human 영상 제작

**CONTENT STEP UP [A.I]**

문화체육관광부 한국콘텐츠진흥원  
KOREA CREATIVE CONTENT AGENCY

교육 장소 한국콘텐츠진흥원 유튜브

10.6(수) 09:20~12:00 STEP 3A. AI 제작 1 이미지/영상



**딥브레인AI 장세영 대표**

(전) SK C&C, 모바일 솔루션 및 메타버스 모바일 개발  
국내 최초 인공지능 앵커 제작  
문재인 대통령 및 손석희 아나운서 음성합성

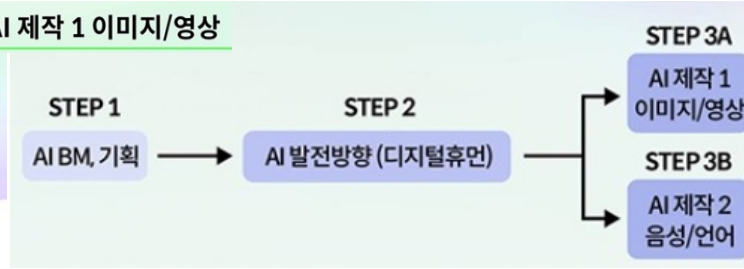
10:15~11:05

**인공인간 (AI Human) 및 버추얼  
인플루언서 비즈니스 혁신**

음성 및 영상 데이터를 활용한 인공인간 기술 분석



김주하 AI앵커



# Contents

## 1. Introduction

### 2. 프로젝트 주제 및 선정 이유

- 정성평가 설문조사

### 3. 데이터 수집 (FSGAN선정 개념)

- Face Swap Model 개념

### 4. 음성 데이터 수집 및 방법

- 데이터 분포 분석
- 음성 추출 방법(VREW)
- FSGAN(얼굴합성)

### 5. 음성 모델 & 모델 성능 비교

- 성능향상을 위한 Pitch(성별)
- Batch size, Class
- CycleGAN VC
- STARGAN VC
- 정량평가(precision, recall, diverge, converge)

## 6. FSGAN 얼굴 합성 한계점

- 인물별 움직임에 따른 성능차이
- Metrics -> Cosine similarity, MSE, SSIM
- 유사도의 따른 합성 결과 (Move, Still)

## 7. 데모영상 시현

## 8. Contribution

- 영상 & 음성 합성 프로세스
- 음성 모델별 비교 성능차이(성별, Pitch)
- 영상 모델별 비교 성능차이

## 9. Future work

# Introduction

## 1. Project Goal

- 음성과 영상 합성을 통한 AI Human 구현하기

## 2. Overall Process (논문리뷰)

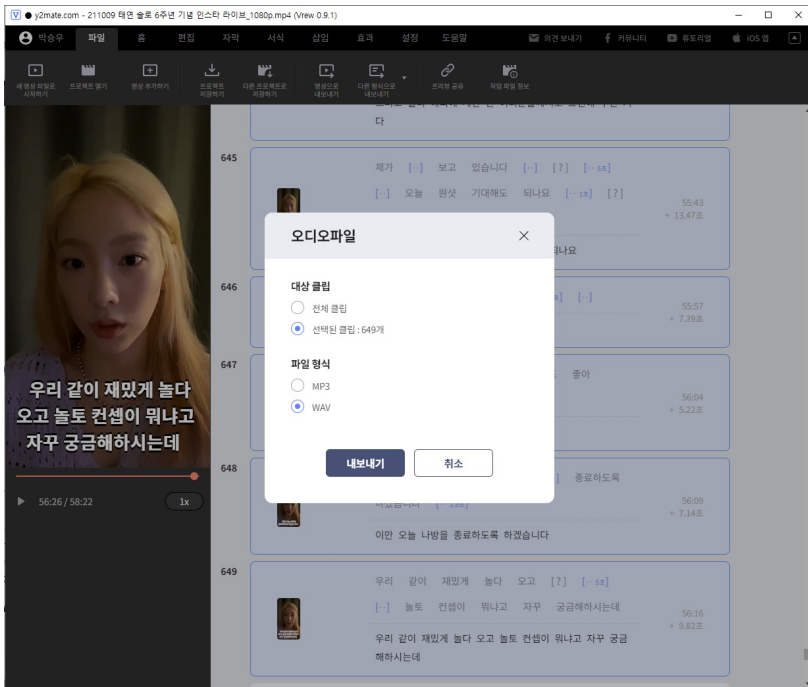
- 논문리뷰
  - 영상: FSGAN
  - 음성: Cyclegan-vc, Stargan-vc, Stargan-vc2
- 코드실습
  - GAN, Cyclegan, Stargan, Fsgan
- 데이터셋 수집
  - 문재인, 김상중, 박근혜, 김주하, 아이유, 유인나
- 모델학습
  - Training: Cyclegan-vc, Stargan-vc, Stargan2-vc
  - Finetuning: Fsgan
- 데모영상
  - 문재인->김상중
  - 박근혜->김주하

## 3. Dataset

### ● Description

- 2 class (문재인, 김상중) – 500개씩
- 4 class (문재인 김상중 박근혜 김주하) - 200개씩
- 4 class ( 김주하, 박근혜, 아이유, 유인나) – 200개씩
- 4 class( 김주하, 박근혜, 아이유, 유인나) – 250개씩
- 500개 단위가 (음성의 한문장)

# Survey



[YouTube to WAV 변환기](#) - Vrew 브루

- 유인나의 불륜을 높여요(1).wav
- 유인나의 불륜을 높여요(2).wav
- 유인나의 불륜을 높여요(3).wav
- 유인나의 불륜을 높여요(4).wav
- 유인나의 불륜을 높여요(5).wav
- 유인나의 불륜을 높여요(6).wav
- 유인나의 불륜을 높여요(7).wav
- 유인나의 불륜을 높여요(8).wav
- 유인나의 불륜을 높여요(9).wav
- 태연\_인스타 라이브\_(1).wav
- 태연\_인스타 라이브\_(2).wav
- 태연\_인스타 라이브\_(3).wav
- 태연\_인스타 라이브\_(4).wav
- 태연\_인스타 라이브\_(5).wav
- 태연\_인스타 라이브\_(6).wav
- 태연\_인스타 라이브\_(7).wav
- 태연\_인스타 라이브\_(8).wav
- 태연\_인스타 라이브\_(9).wav
- 태연\_인스타 라이브\_(10).wav
- 태연\_인스타 라이브\_(11).wav
- 태연\_인스타 라이브\_(12).wav

한문장별로 나눠  
음성 데이터 취합

<https://forms.gle/ke3b8qgeZACYCJSm6>

Voice conversion evaluation

다음 음성 중 [문재인 대통령](#)과 가장 비슷한 목소리 음성을 선택해 주세요. (File responses)

Q1-2: link \*

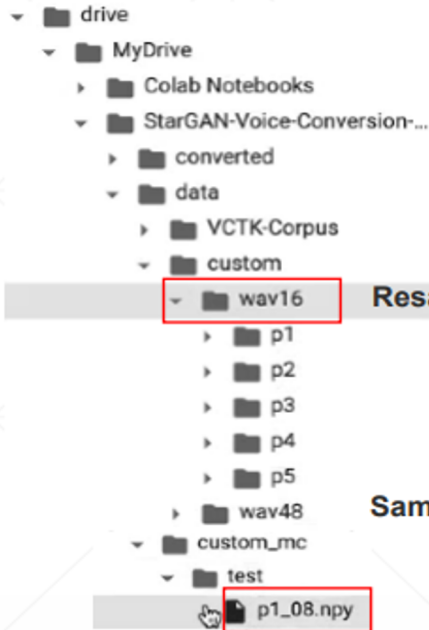
<https://drive.google.com/drive/folders/1VJEvw0HVvVsK9PCW6LLrMj6piUjGEyY?usp=sharing>

	오디오-1	오디오-2	오디오-3	오디오-4
1st	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
2nd	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
3rd	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
4th	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

음성 합성된 음성 정성평가분석을  
위한 설문조사 진행

# StarGAN VC source analysis

## StarGan-VC 소스 분석



Resampling 16k

Sampling 48k

남자 인물

음성file -> Feature vector

GPU check

```
import torch
torch.cuda.is_available()
True GPU 확인
```

```
preprocess.py X
44 train_paths = list(np.array(paths)[train_indices])
45 test_paths = list(np.array(paths)[test_indices])
46 return train_paths, test_paths
47
48 def get_spk_world_feats(spk_fold_path, mc_dir_train, mc_dir_test, sample_rate=16000):
49     paths = glob.glob(join(spk_fold_path, '*.wav'))
50     spk_name = basename(spk_fold_path)
51     train_paths, test_paths = split_data(paths)
52     f0s = []
53     coded_sps = []
54     for wav_file in train_paths:
55         f0, _, _, _, coded_sp = world_encode_wav(wav_file, fs=sample_rate)
56         f0s.append(f0)
57         coded_sps.append(coded_sp)
58     log_f0s_mean, log_f0s_std = logf0_statistics(f0s)
59     coded_sps_mean, coded_sps_std = coded_sp_statistics(coded_sps)
60     np.savez(join(mc_dir_train, spk_name+'_stats.npz'),
61             log_f0s_mean=log_f0s_mean,
62             log_f0s_std=log_f0s_std,
63             coded_sps_mean=coded_sps_mean,
64             coded_sps_std=coded_sps_std)
65
66 for wav_file in tqdm(test_paths):
67     wav_name = basename(wav_file)
68     f0, timeaxis, sp, ap, coded_sp = world_encode_wav(wav_file, fs=sample_rate)
69     normed_coded_sp = normalize_coded_sp(coded_sp, coded_sps_mean, coded_sps_std)
70     np.save(join(mc_dir_test, wav_name.replace('.wav', '.npy')), normed_coded_sp)
71 return 0
```



Generator 학습

Run Demo

Iteration 확인

```
[ ] !python convert.py --resume iters 38000 --src_spk p3 --trg_spk p2
```

인물3 -> 인물2

p1, p2, p3, p4 = 발라드, 일연비, 트로트, 어쿠스

p1, p2, p3, p4, p5

```
return torch.FloatTensor(mc), torch.LongTensor([spk_idx]).squeeze_(), torch.FloatTensor(spk_cat)
```

전처리 결과

Speaker Index화

Speaker Category화

1% -> 13분 소요

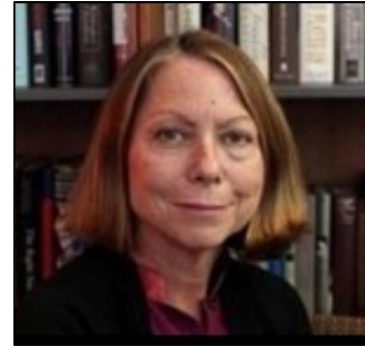
Train Model

```
!python main.py
Elapsed [0:11:27], Iteration [870/100000], D/loss_real: -36.1231, D/loss_fake: 6.6896, D/loss_cls_spks: 1.1928, D/loss_gp
Elapsed [0:11:34], Iteration [880/100000], D/loss_real: -32.9494, D/loss_fake: 3.8188, D/loss_cls_spks: 1.1157, D/loss_gp
Elapsed [0:11:43], Iteration [890/100000], D/loss_real: -8.3074, D/loss_fake: -20.2896, D/loss_cls_spks: 1.3163, D/loss_gp
Elapsed [0:11:50], Iteration [900/100000], D/loss_real: -38.8917, D/loss_fake: 12.8880, D/loss_cls_spks: 1.1065, D/loss_gp
Elapsed [0:11:57], Iteration [910/100000], D/loss_real: -39.9694, D/loss_fake: 9.1933, D/loss_cls_spks: 1.1478, D/loss_gp
Elapsed [0:12:04], Iteration [920/100000], D/loss_real: -38.6882, D/loss_fake: 8.3282, D/loss_cls_spks: 0.9823, D/loss_gp
Elapsed [0:12:13], Iteration [930/100000], D/loss_real: -36.9822, D/loss_fake: 11.2652, D/loss_cls_spks: 1.5355, D/loss_gp
Elapsed [0:12:22], Iteration [940/100000], D/loss_real: -65.2353, D/loss_fake: 28.0413, D/loss_cls_spks: 1.0007, D/loss_gp
Elapsed [0:12:29], Iteration [950/100000], D/loss_real: -33.5546, D/loss_fake: 3.7970, D/loss_cls_spks: 1.2533, D/loss_gp
Elapsed [0:12:37], Iteration [960/100000], D/loss_real: -49.9136, D/loss_fake: 22.5645, D/loss_cls_spks: 0.9133, D/loss_gp
Elapsed [0:12:45], Iteration [970/100000], D/loss_real: -31.6537, D/loss_fake: 6.8243, D/loss_cls_spks: 0.9622, D/loss_gp
Elapsed [0:12:52], Iteration [980/100000], D/loss_real: -22.6087, D/loss_fake: -8.8156, D/loss_cls_spks: 1.0101, D/loss_gp
Elapsed [0:13:01], Iteration [990/100000], D/loss_real: -54.6807, D/loss_fake: 20.8980, D/loss_cls_spks: 1.3844, D/loss_gp
Elapsed [0:13:08], Iteration [1000/100000], D/loss_real: -56.6622, D/loss_fake: 27.0038, D/loss_cls_spks: 0.7710, D/loss_gp
```

# Face Swap

## Face Swap 기술이란?

사진 혹은 비디오에 있는 얼굴을 인식하여, 얼굴의 부분을 우리가 원하는 대상의 얼굴 부분으로 바꿔주는 딥러닝 기반의 기술을 의미한다.



Source

+



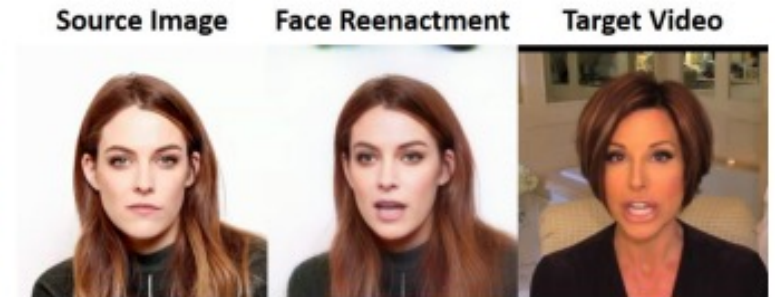
Target



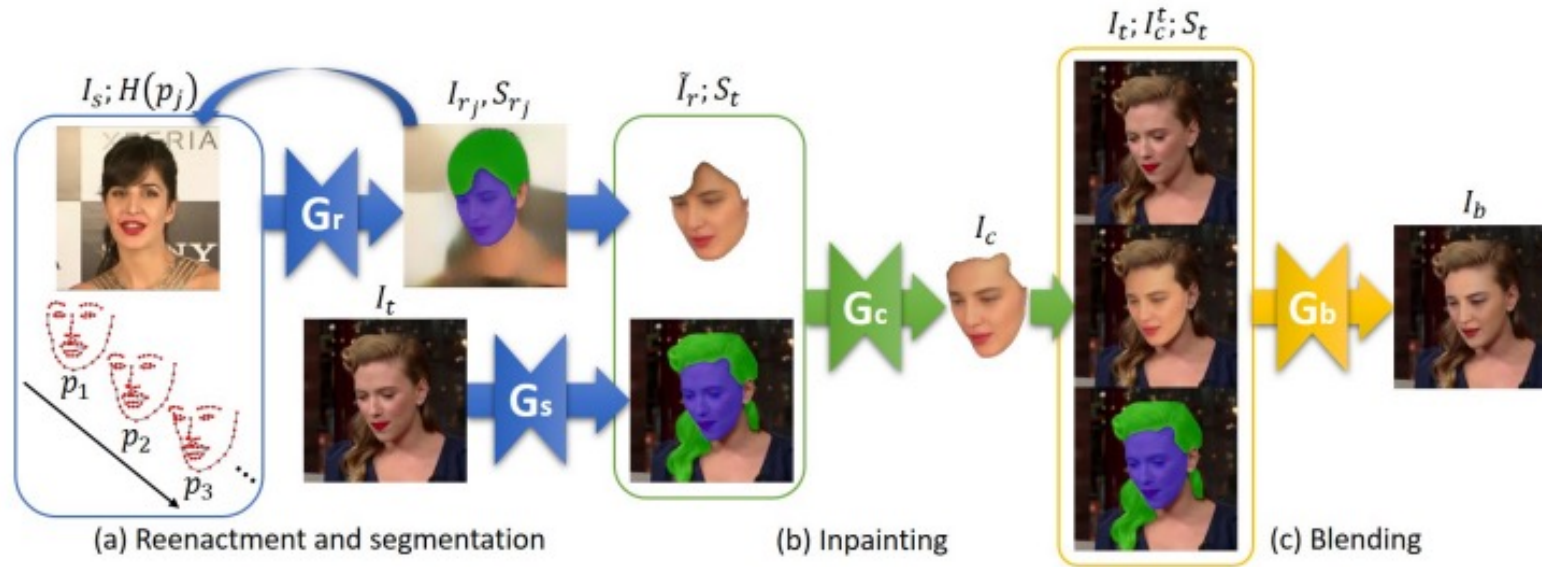
Result

## FSGAN (Face-Swapping Generative Adversarial Network)

Goal: Subject-agnostic face swap



# FSGAN(Face Swapping Generative Adversarial Network)



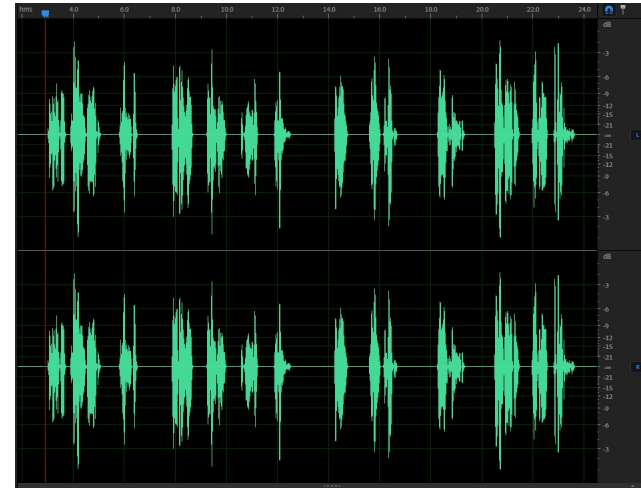
Contribution:

1. RNN기반으로 접근하여 사진 한 장 또는 영상 시퀀스에 적용하여 얼굴 재현을 가능하게 함.
2. Face completion network를 사용하여 얼굴 영역을 생성하고, face blending network를 통해 target의 얼굴색과 조명색을 유지하여 두 얼굴이 매끄럽게 합성되도록 함.
3. 새로운 손실함수인 Poisson blending loss를 사용.



# Voice preprocessing

- 고품질의 음성 데이터 수집 및 음성 파일 preprocessing
  - Prototype #1 문재인과 김상중 간 목소리 변환 모델의 데이터 500여개 수집

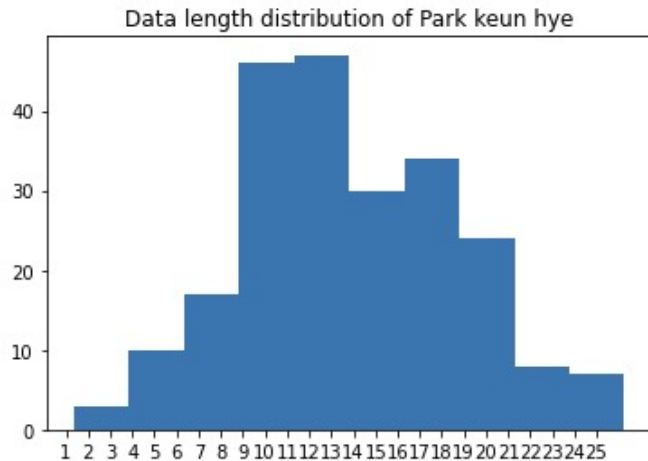


- Prototype #2 여성 화자 기반의 목소리 데이터 200여개 파일 수집

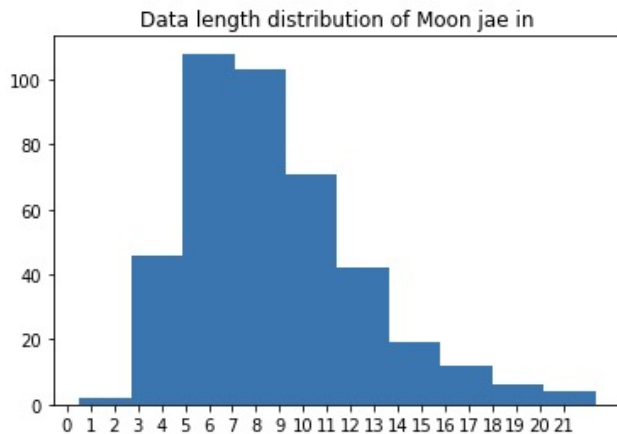
# 팀 내 역할과 주별 주요 활동

- 음성 데이터 분포 분석

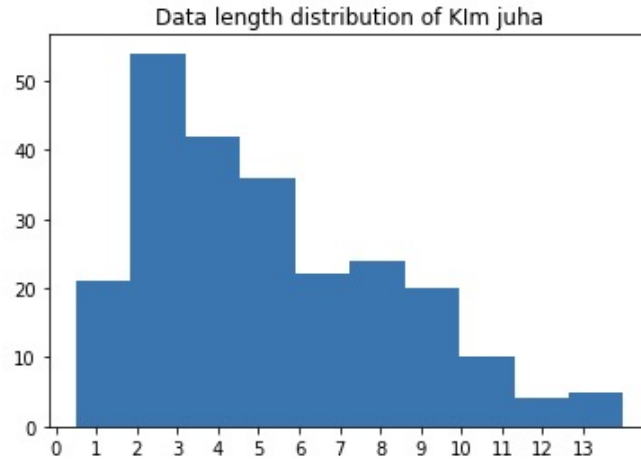
mean value: 13.9025958702065  
sample\_n: 226



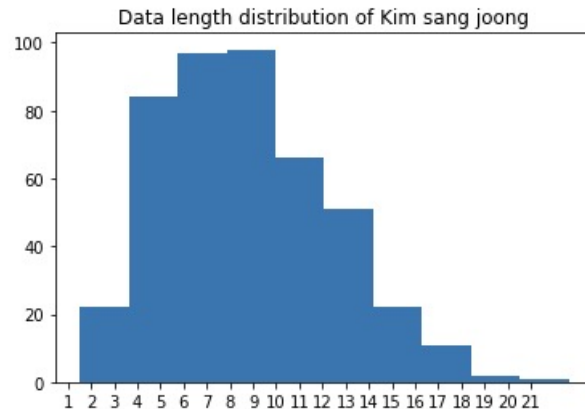
mean value: 8.746853056900717  
sample\_n: 413



mean value: 5.263165266106442  
sample\_n: 238



mean value: 8.778942731277533  
sample\_n: 454



```
import os

DIR = "/content/drive/MyDrive/StarGAN-Voice-Conversion-master/"

files = os.listdir(DIR)
files.sort()

lengDict=dict()
for target_id in files[0:]:
    print(target_id)
    subDir=os.listdir(DIR+str(target_id))
    subDir.sort()
    lengDict[target_id]=[]
    for num,file in enumerate(subDir):
        if num%20 ==0:
            print("file :",file)
            f = sf.SoundFile(DIR+str(target_id)+"/"+str(file))

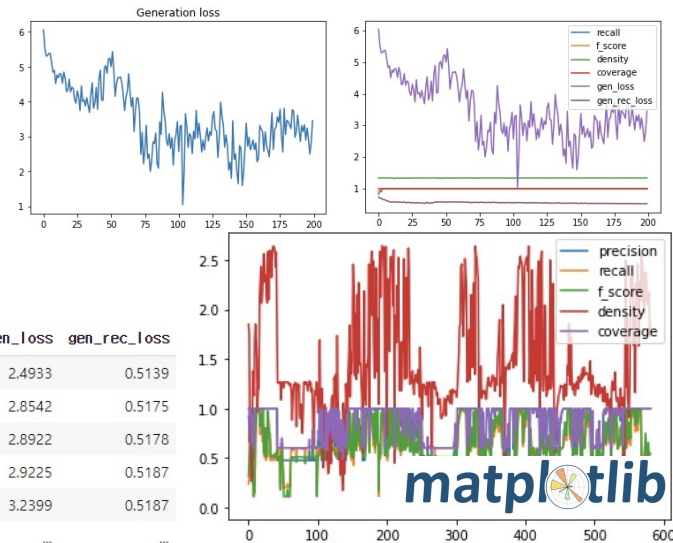
            lengDict[target_id].append(f.frames / f.samplerate)

print("mean value: ",sum(lengDict['pl'])/len(lengDict['pl']))
plt.hist(lengDict['pl'])
print("sample_n: ",len(lengDict['pl']))
x=list(range(int(min(lengDict['pl'])),int(max(lengDict['pl'])))
plt.xticks(x)
plt.title("Data length distribution of Kim sang joong")
plt.show()

pl
file : pl_01.wav
file : pl_11.wav
file : pl_128.wav
```

# 팀 내 역할과 주별 주요 활동

- 학습성능 향상을 위한 영상 편집 및 마스크 처리
- FSGAN
  - 박근혜 김주하 얼굴 합성 및 파라미터 튜닝
- starGAN
  - Model training
  - Loss terms, converge and density 등의 정량 평가 데이터 분석, 가시화 및 최적 모델 탐색
- 데모 시연 영상 편집 및 제작



```
# Utility functions
import ffmpeg

def encode_audio(video_path, audio_path, output_path):
    ffmpeg.concat(ffmpeg.input(video_path), ffmpeg.input(audio_path), output=output_path, strict='-2').run(overwrite_output=True)

def display_video(video_path, width=640, height=480):
    vid_data = open(video_path, 'rb').read()
    vid_url = 'data:video/mp4;base64,' + b64encode(vid_data).decode()

    if clear:
        clear_output()

    return HTML(f"""
    <video width={width} height={height}>
    <source src={vid_url} type="video/mp4">
    </video>
    """)

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

Drive already mounted at /content/drive: to access files, use:
!rm -dr /content/data/source*
!rm -dr /content/data/target*
```

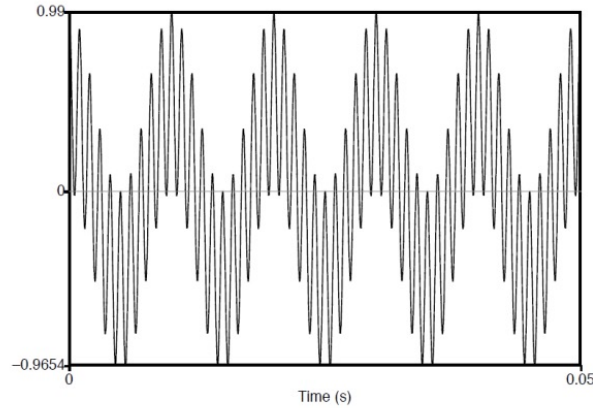
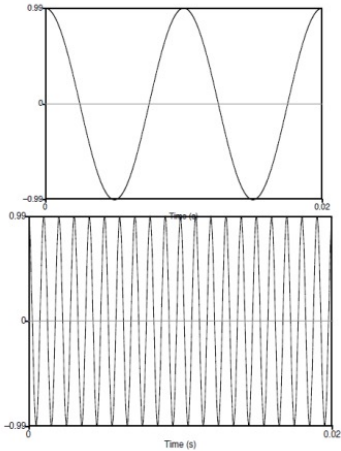
Elapsed	Iteration	D/loss_real	D/loss_fa	iter	precision	recall	f_score	density	coverage	gen_loss	gen_rec_loss	
Elapsed [0:07:07]	Iteration [1000/145000]	D/loss_real: -107.6250	D/loss_fa: -39.7920	...	...	...	...	...	...	...	...	
Elapsed [0:14:34]	Iteration [2000/145000]	D/loss_real: -22.3380	D/loss_fa: -39.7920	197	198000/200000	1.0000	1.0000	1.0000	1.3333	1.0	2.4933	0.5139
Elapsed [0:21:59]	Iteration [3000/145000]	D/loss_real: -39.7920	D/loss_fa: -54.0177	190	191000/200000	1.0000	1.0000	1.0000	1.3307	1.0	2.8542	0.5175
Elapsed [0:29:25]	Iteration [4000/145000]	D/loss_real: -54.0177	D/loss_fa: -49.2912	194	195000/200000	1.0000	1.0000	1.0000	1.3333	1.0	2.8922	0.5178
Elapsed [0:36:50]	Iteration [5000/145000]	D/loss_real: -49.2912	D/loss_fa: -55.7255	196	197000/200000	1.0000	1.0000	1.0000	1.3333	1.0	2.9225	0.5187
Elapsed [0:44:16]	Iteration [6000/145000]	D/loss_real: -55.7255	D/loss_fa: -55.7255	189	190000/200000	1.0000	1.0000	1.0000	1.3333	1.0	3.2399	0.5187
Elapsed [0:51:42]	Iteration [7000/145000]	D/loss_real: -55.7255	D/loss_fa: -55.7255	...	...	...	...	...	...	...	...	
Elapsed [0:59:08]	Iteration [8000/145000]	D/loss_real: -55.7255	D/loss_fa: -55.7255	...	...	...	...	...	...	...	...	
Elapsed [1:06:34]	Iteration [9000/145000]	D/loss_real: -55.7255	D/loss_fa: -55.7255	...	...	...	...	...	...	...	...	
Elapsed [1:13:59]	Iteration [10000/145000]	D/loss_real: -55.7255	D/loss_fa: -55.7255	...	...	...	...	...	...	...	...	



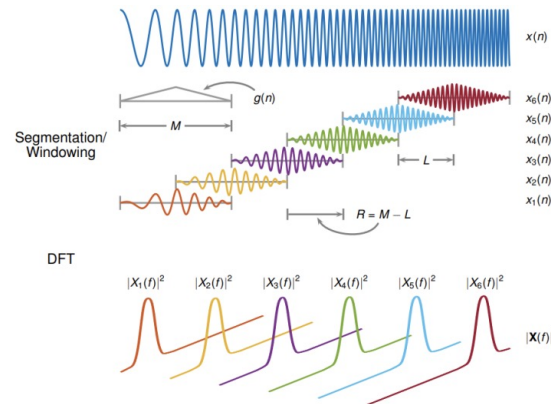
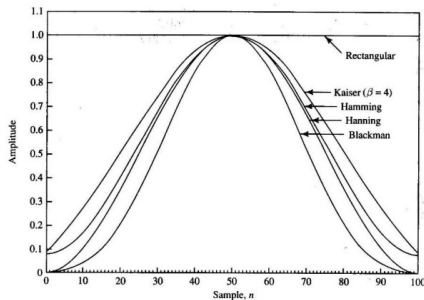
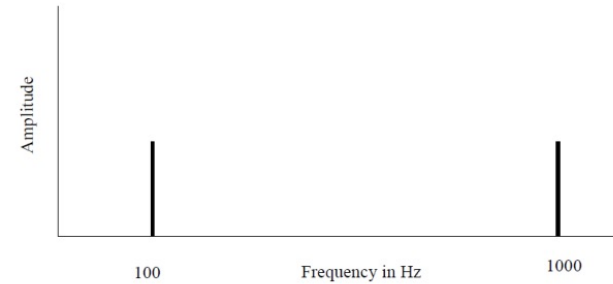
# 음성 feature 추출 방법

## Voice data 분석 및 특징 추출법 연구

### 음성 파일 추출과 푸리에 변환 코드화



Frequency components (100 and 1000 Hz) on x-axis



```
def get_spk_world_feats(spk_fold_path, mc_dir_train, mc_dir_test):
    paths = glob.glob(join(spk_fold_path, '*.wav'))
    spk_name = basename(spk_fold_path)
    train_paths, test_paths = split_data(paths)
    f0s = []
    coded_sps = []
    for wav_file in train_paths:
        f0, timeaxis, coded_sp = world_encode_wav(wav_file, fs, frame_period)
        f0s.append(f0)
        coded_sps.append(coded_sp)
    log_f0s_mean, log_f0s_std = logf0_statistics(f0s)
    coded_sps_mean, coded_sps_std = coded_sp_statistics(coded_sps)
    np.savez(join(mc_dir_train, spk_name+'_stats.npz'),
             log_f0s_mean=log_f0s_mean,
             log_f0s_std=log_f0s_std,
             coded_sps_mean=coded_sps_mean,
             coded_sps_std=coded_sps_std)

    for wav_file in tqdm(train_paths):
        wav_name = basename(wav_file)
        f0, timeaxis, sp, ap, coded_sp = world_encode_wav(wav_file, fs, frame_period)
        normed_coded_sp = normalize_coded_sp(coded_sp, coded_sps_mean, coded_sps_std)
        np.save(join(mc_dir_train, wav_name.replace('.wav', '.npy')))

def world_decompose(wav, fs, frame_period = 5.0):
    # Decompose speech signal into f0, spectral envelope and aperiodic part
    wav = wav.astype(np.float64)
    f0, timeaxis = pyworld.harvest(wav, fs, frame_period)
    sp = pyworld.cheaptrick(wav, f0, timeaxis, fs)
    ap = pyworld.d4c(wav, f0, timeaxis, fs)
    return f0, timeaxis, sp, ap

def world_encode_spectral_envelop(sp, fs, dim=36):
    # Get Mel-cepstral coefficients (MCEPs)
    # sp = sp.astype(np.float64)
    coded_sp = pyworld.code_spectral_envelope(sp, fs, dim)
    return coded_sp

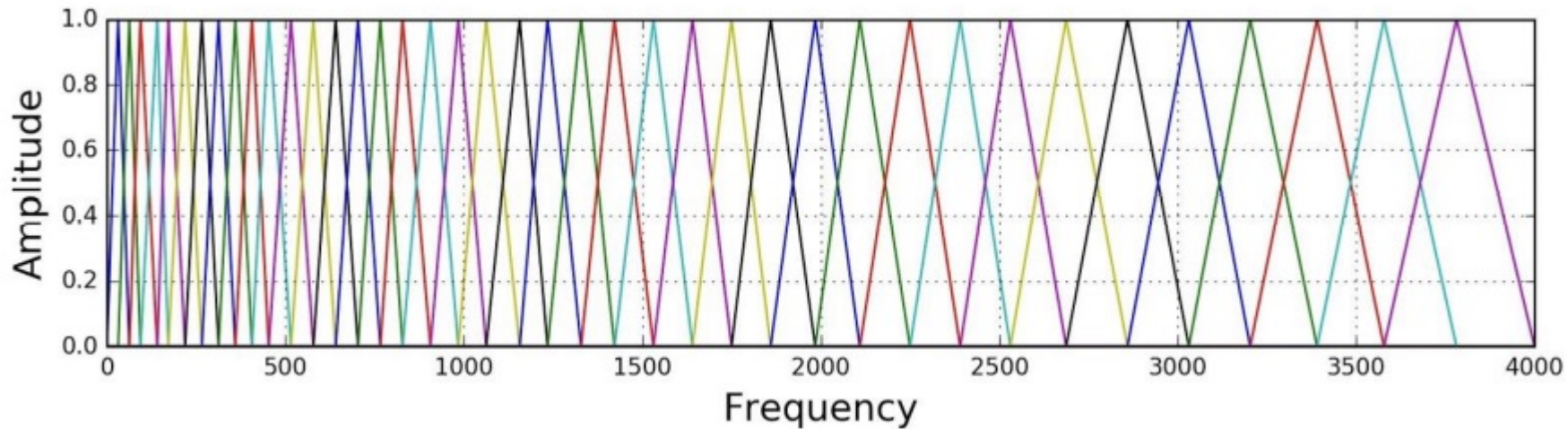
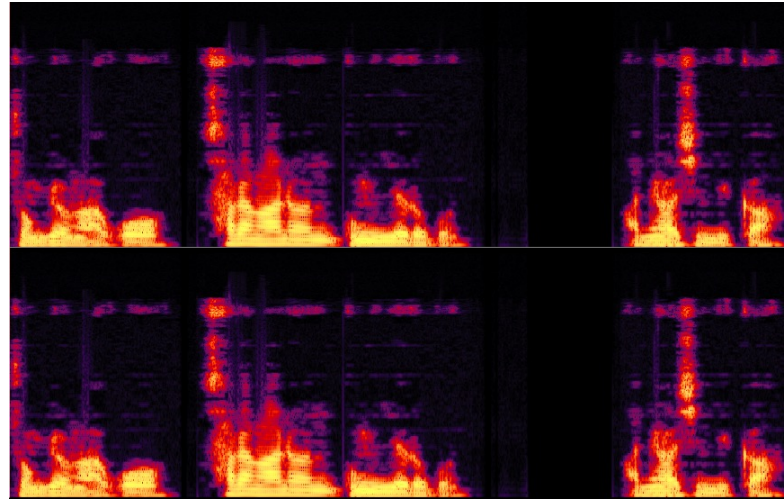
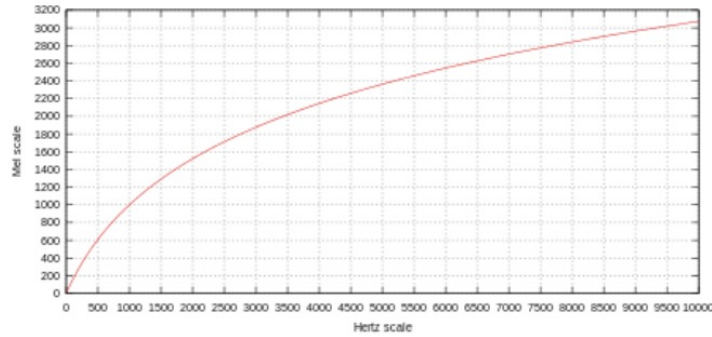
def world_decode_spectral_envelop(coded_sp, fs):
    # Decode Mel-cepstral to sp
    fftlen = pyworld.get_cheaptrick_fft_size(fs)
    decoded_sp = pyworld.decode_spectral_envelope(coded_sp, fs, fftlen)
    return decoded_sp

def world_encode_wav(wav_file, fs, frame_period=5.0):
    wav = load_wav(wav_file, sr=fs)
    f0, timeaxis, sp, ap = world_decompose(wav=wav, fs=fs, frame_period=frame_period)
    coded_sp = world_encode_spectral_envelop(sp, fs, dim=36)
    return f0, timeaxis, sp, ap, coded_sp
```

# 음성 feature 추출 방법

## Voice data 분석 및 특징 추출법 연구

### ▪ MCEP 채택



```
return wav_padded

def logf0_statistics(f0s):
    log_f0s_concatenated = np
    log_f0s_mean = log_f0s_co
    log_f0s_std = log_f0s_con

    return log_f0s_mean, log_

def pitch_conversion(f0, mean

# Logarithm Gaussian norm
f0_converted = np.exp((np

return f0_converted

def wavs_to_specs(wavs, n_fft

stfts = list()
for wav in wavs:
    stft = librosa.stft(w
    stfts.append(stft)

return stfts
```

# FSGAN

얼굴 합성 데모 영상 생성 (박근혜 <-> 김주하)

->결과가 좋지 않아 fsGAN으로 모델 변경

- fsGAN 코드 실행 및 데모 생성

->결과 확인을 반복하여 가용 resource내에서

파라미터 튜닝

Iteration 800 > 1000, batch size 8 > 30



```
import os
from fsgan.inference.swap import FaceSwapping
from fsgan.criterions.vgg_loss import VGGLoss

#@markdown This step should only be done once unless one of the
#@markdown following parameters is changed:

#@markdown ---
#@markdown Path to the weights directory (make sure it is corre
weights_dir = '/content/drive/My Drive/fsgan/weights' #@param
#@markdown Number of finetune iterations on the source subject
finetune_iterations = 1000 #@param {type:"slider", min:100, ma
#@markdown If True, the inner part of the mouth will be remove
seg_remove_mouth = True #@param {type:"boolean"}
#@markdown Segmentation batch size
seg_batch_size = 30 #@param {type:"slider", min:1, max:64, ste
#@markdown Inference batch size
batch_size = 30 #@param {type:"slider", min:1, max:64, step:1}
#@markdown ---

detection_model = os.path.join(weights_dir, 'v2/WIDERFace_DSFD
pose_model = os.path.join(weights_dir, 'shared/hopenet_robust_
lms_model = os.path.join(weights_dir, 'v2/hr18_wflw_landmarks.t
seg_model = os.path.join(weights_dir, 'v2/celeba_unet_256_1_2_3
reenactment_model = os.path.join(weights_dir, 'v2/nfv_msrunet_3
completion_model = os.path.join(weights_dir, 'v2/ijbc_msrunet_3
blending_model = os.path.join(weights_dir, 'v2/ijbc_msrunet_256
criterion_id_path = os.path.join(weights_dir, 'v2/vggface2_vgg
criterion_id = VGGLoss(criterion_id_path)
```

# Voice Conversion

: 화자가 전달하려는 내용 (linguistic content)은 유지하고 우리가 원하는 대상화자의 말하기 특징 (acoustic feature)을 합성하여 목소리 스타일을 변환하는 기술을 의미한다.

## 발화 내용

- 동일한 발화 데이터를 사용하는지 여부 – Parallel vs Non-Parallel
- 발화된 언어가 동일한지 여부 – Mono-lingual vs Cross-lingual

## 발화자 구성

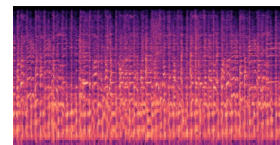
- 여러 명의 화자를 사용하는지 여부 – (one-to-one) vs (many-to-many)
- 동일한 성별의 화자를 사용하는지 여부 – (same gender) vs (different gender)



source speaker

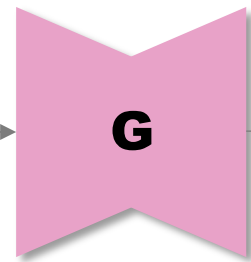
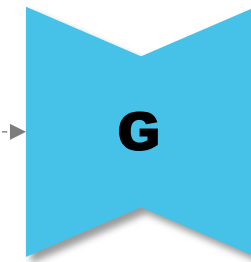


waveform audio



mel-spectrogram

$x$



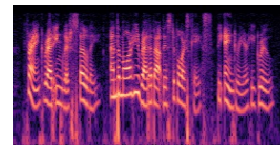
$\hat{x}$



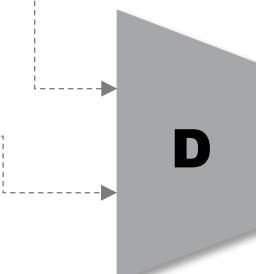
target speaker



waveform audio



mel-spectrogram



Real/Fake

# Voice Conversion Models

## 모델 선택

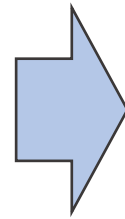
: CycleGAN-VC, StarGAN-VC

## 목적

: 각 모델이 갖는 특성에 따라 합성된 음성의 퀄리티가 어떻게 달라지는지 살펴보고 음성 합성 도메인에서 개선의 여지가 있는 기능들을 살펴보고자 한다.

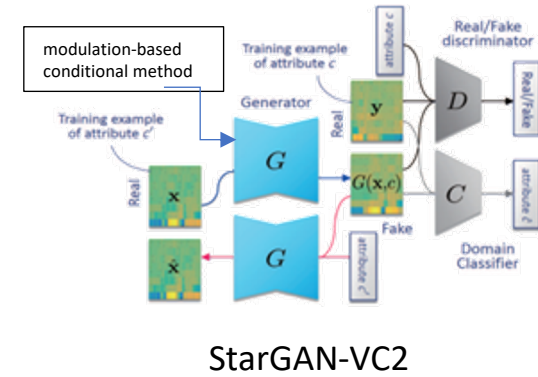
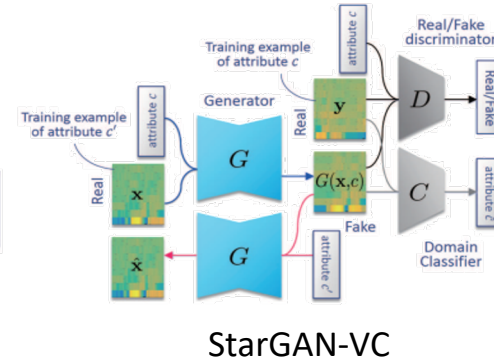
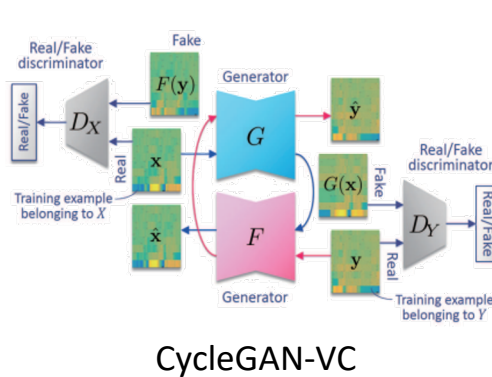
## 선행 분석

- 다양한 sampling rate 설정 : 16K, 22K, 32K
- 배치사이즈 변경 : 16, 128, 256
- 학습 iteration : 50K, 100K, 200K
- 다양한 목소리 특징을 갖는 화자를 구성하여 학습 : 박근혜, 유인나, 아이유, 김주하



## 분석 결과

- 기존 연구에서 사용되는 16K로 resampling하는 결과가 제일 좋음
- 배치사이즈가 적을 수록 미세하게 좋은 합성 결과를 얻음
- 배치사이즈를 작게 설정하고 iteration을 많이 설정하면 비교적 안정적인 음성 합성 결과를 얻음
- 목소리의 높낮이나 말의 빠르기가 많이 차이나는 화자 간의 음성합성은 잘 되지 않는 결과를 얻음





# Model Comparison (1)

	CycleGAN-VC	StarGAN-VC	StarGAN-VC	StarGAN-VC2
Classification Loss (discriminator)	×	0	0	0
Domain code modulation (Generator)	×	Depth-wise modulation	Depth-wise modulation	Conditional instance normalization
Reconstruction Loss	0	0	0	0
Domain A-Domain B	One-to-One	One-to-One	Many-to-Many	Many-to-Many

CycleGAN-VC (one-to-one)

VS

StarGAN-VC (one-to-one)



StarGAN-VC (one-to-one)

VS

StarGAN-VC (Many-to-Many)



```

Stargan-VC(four speakers).ipynb
파일 수정 보기 삽입 런타임 도구 도움말 10월 29일에 마지막으로 저장됨

+ 코드 + 텍스트

Move to repository in my GoogleDrive

[ ] cd drive/MyDrive/StarGAN-Voice-Conversion-master
/content/drive/MyDrive/StarGAN-Voice-Conversion-master

Preprocess data using Mel-cepstral coefficients(MCEPs) (already done)

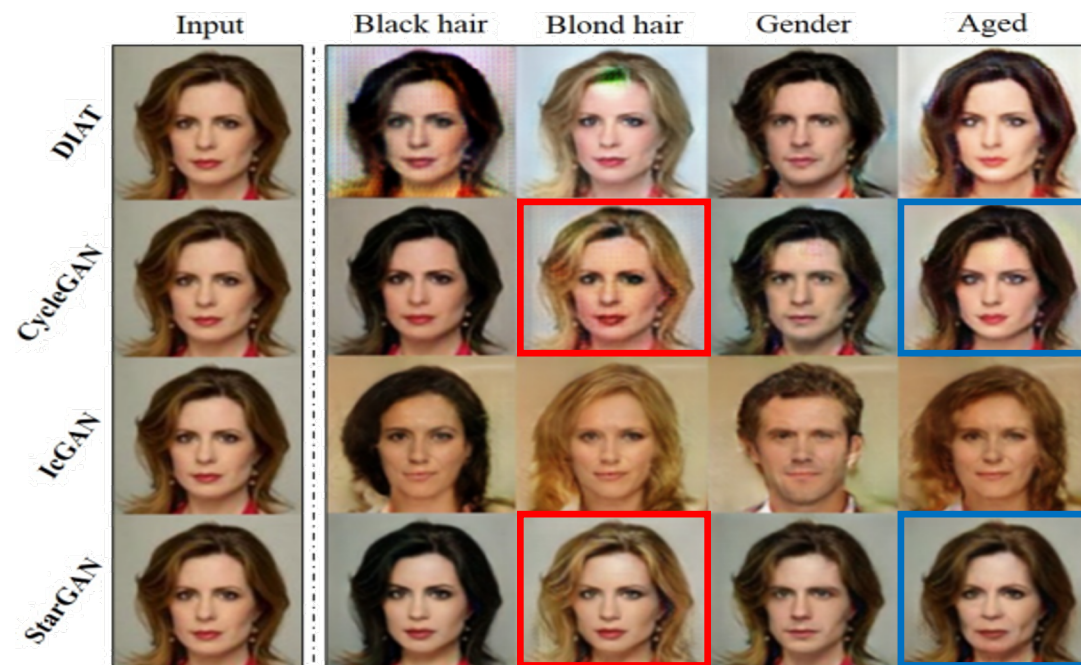
[ ] !python preprocess.py --sample_rate 16000 \
--train_wavpath data/custom/wav48 \
--target_wavpath data/custom/wav16 \
--mc_dir_train data/custom_mc/train \
--mc_dir_test data/custom_mc/test

GPU check

[ ] import torch
torch.cuda.is_available()
    
```

```

MyDrive
├── Colab Notebooks
├── StarGAN-Voice-Conversion-...
│   ├── converted
│   ├── converted_samples
│   ├── data
│   ├── logs
│   ├── models
│   ├── models(bsz128)
│   ├── samples
│   ├── samples(bsz128)
│   ├── samples(error)
│   ├── README.md
│   ├── convert.py
│   ├── data_loader.py
│   ├── logger.py
│   ├── main.py
│   └── model.py
├── converted
├── converted_samples
├── data
│   ├── custom
│   ├── custom_mc
│   ├── demo_feats
│   ├── demo_wav16
│   ├── my_sample
│   ├── my_sample_mc
│   └── speaker-info.txt
├── logs
│   ├── test_logs(bsz128).txt
│   ├── test_logs.txt
│   ├── train_logs(bsz128).txt
│   └── train_logs.txt
├── models
├── models(bsz128)
└── samples
    
```



Facial attribute transfer results on StarGAN

# Model Comparison (2)

	CycleGAN-VC	StarGAN-VC	StarGAN-VC	StarGAN-VC2
Classification Loss (discriminator)	×	0	0	0
Domain code modulation (Generator)	×	Depth-wise modulation	Depth-wise modulation	Conditional instance normalization
Reconstruction Loss	0	0	0	0
Domain A-Domain B	One-to-One	One-to-One	Many-to-Many	Many-to-Many

## StarGAN-VC (Many-to-Many) vs StarGAN-VC2 (Many-to-Many)



```

Run Demo
▶ ipython convert.py --resume_iters 100000 --src_spk p1 --trg_spk p2
↳ Namespace(convert_dir='./converted', log_dir='./logs', model_save_dir='./models', num_converted_wav=1)
Loading the trained models from step 100000...
131120
Before being fed into G: (1640, 36)
After being fed into G: (1640, 36)
112560
Before being fed into G: (1408, 36)
After being fed into G: (1408, 36)
67760
Before being fed into G: (848, 36)
After being fed into G: (848, 36)
65840
Before being fed into G: (824, 36)
After being fed into G: (824, 36)
177200
Before being fed into G: (2216, 36)
After being fed into G: (2216, 36)
159920
Before being fed into G: (2000, 36)
After being fed into G: (2000, 36)
193200
Before being fed into G: (2416, 36)
After being fed into G: (2416, 36)
140720
Before being fed into G: (1760, 36)

```

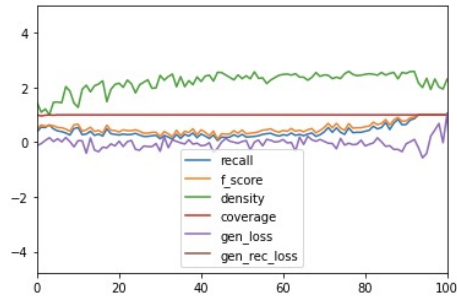
### Limitation on previous methods

: Source speaker의 음성 길이에 맞춰 target 스피커의 말 하기 특징 (acoustic feature)을 합성하다 보니 여전히 source speaker의 음성 특징 정보가 남아 음성합성 결과에 안 좋은 영향을 미침

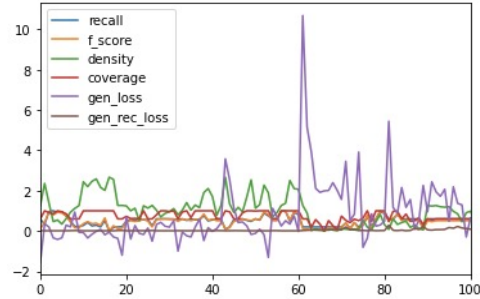
# Experimental Results (1)

## 정량적 결과 (Quantitative Results)

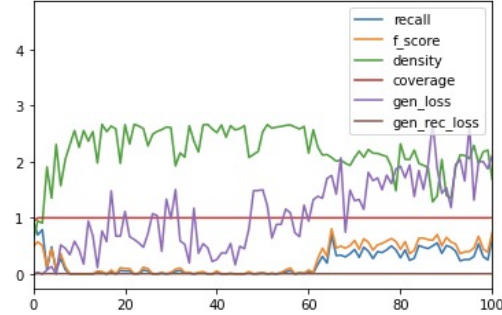
Evaluation metrics: precision, recall, density, coverage



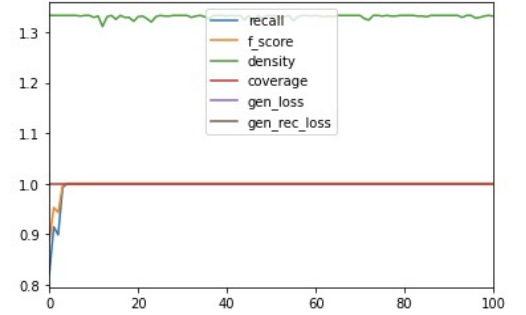
CycleGAN-VC



StarGAN-VC (2p)



StarGAN-VC (4p)



StarGAN-VC 2 (4p)

## 정성적 결과 (Qualitative Results)

평가 방법: 구글 설문조사를 통해 음성파일을 듣고 좀 더 문재인 (target speaker)의 목소리와 비슷한 샘플에 순위를 매기도록 함

음성 합성 샘플 : 김상중 → 문재인

설문조사자 수: 25명

질문 목록

- 각 모델이 합성한 음성 샘플을 비교하여 어떤 모델이 정성적으로 좋은 결과 가지는가?
- 정량지표 (Density, Reconstruction Loss, Generation Loss) 의 min, max, median 값을 시점 (iteration)의 모델의 정성평가 (StarGAN-VC 2 모델로 고정)

# Experimental Results (2)

Model A : CycleGAN-VC

Model B : StarGAN-VC (2p)

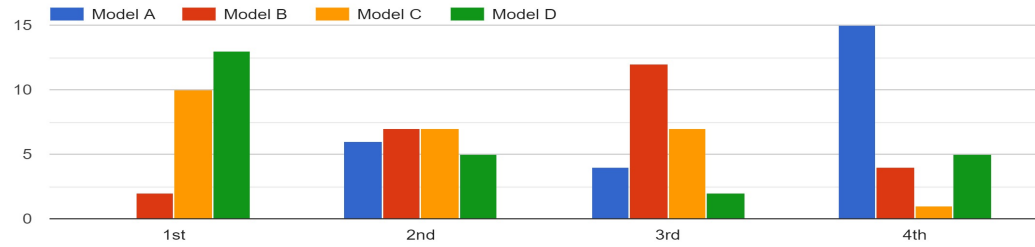
Model C : StarGAN-VC (4p)

Model D : StarGAN-VC 2 (4p)

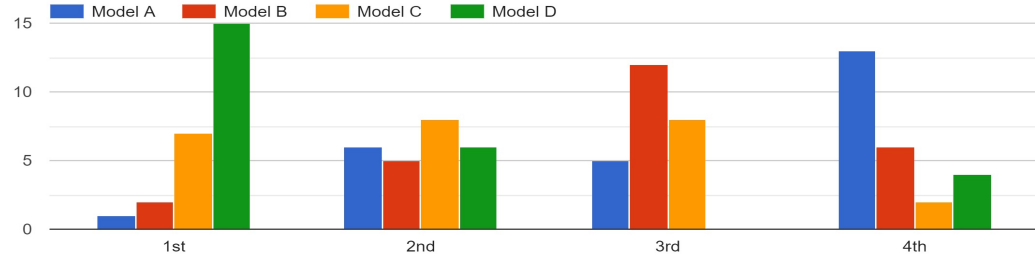
Recon\_loss: reconstruction loss

Gen\_loss : generation loss

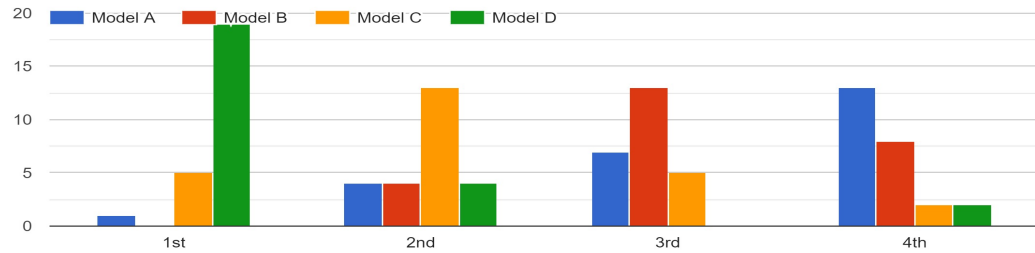
Q1-1



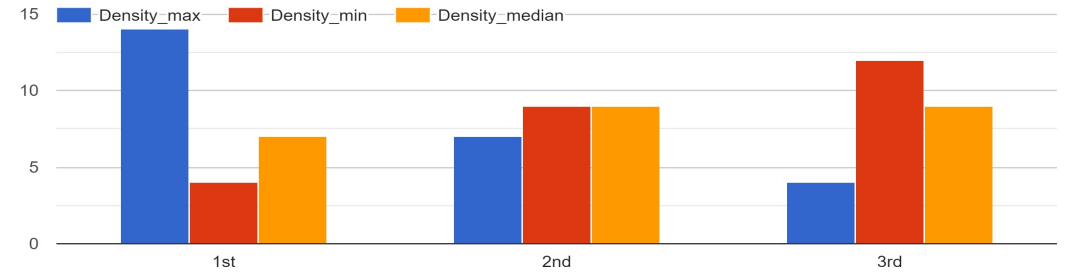
Q1-2 : link



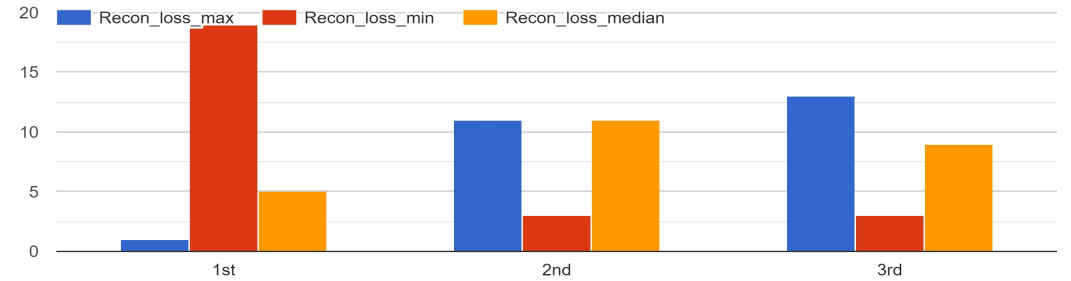
Q1-3 : link



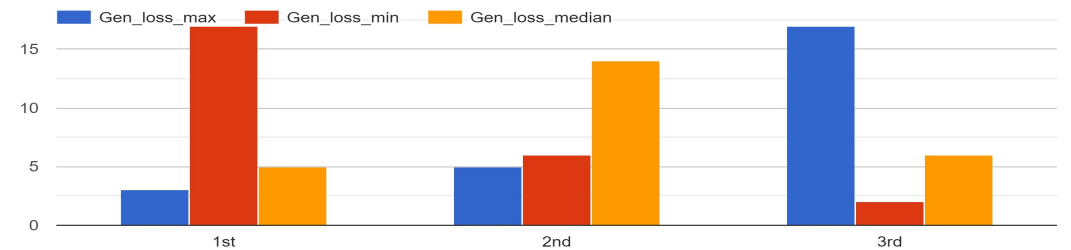
Q2-1 : link



Q2-2 : link



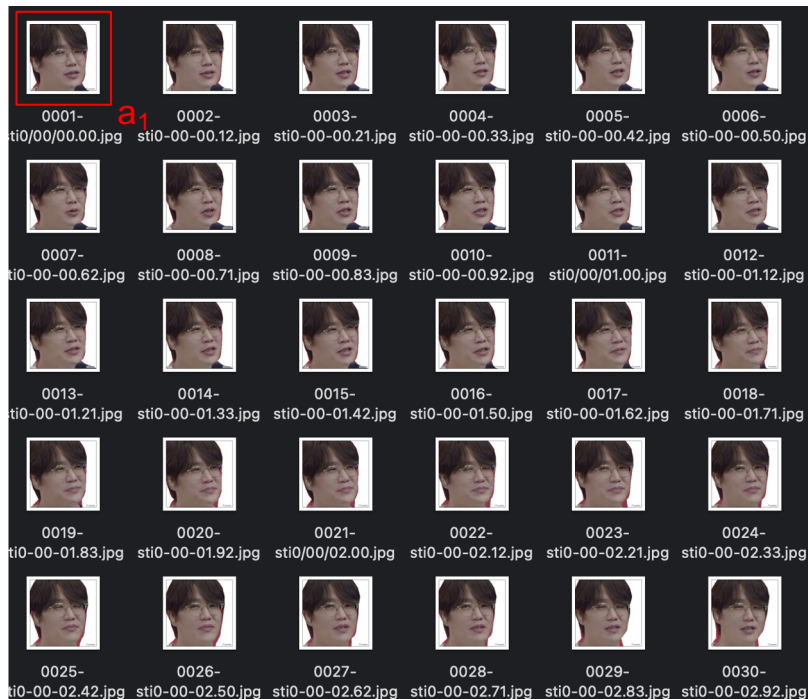
Q2-3 : link



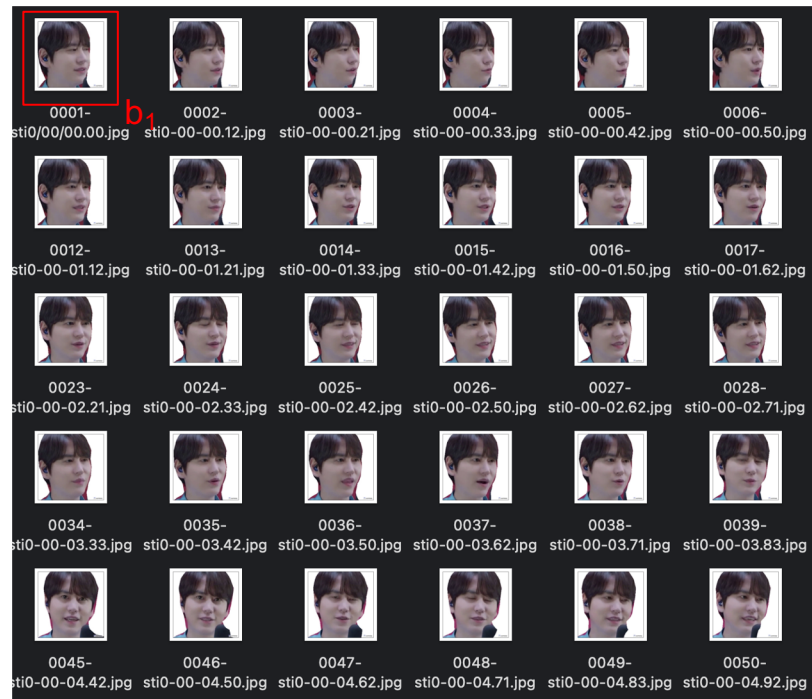
# Average similarity

- Problem
  - The more movement on the face, the lower the performance
  - It takes about an hour to synthesize the two videos.
- Average similarity between the two videos
  - Feature types: Pixel vs. Face embedding(FaceNet) vs. PCA
  - Metrics: Cosine similarity, MSE, SSIM(Structural Similarity Index Measure)
  - $\frac{\Sigma[Cosine\ sim, MSE, SSIM](a_n, b_n)}{n}$

A



B



# Average similarity



Cosine sim: 0.7563 | 비 → 김종국



Cosine sim: 0.8391 | 성시경 → 비

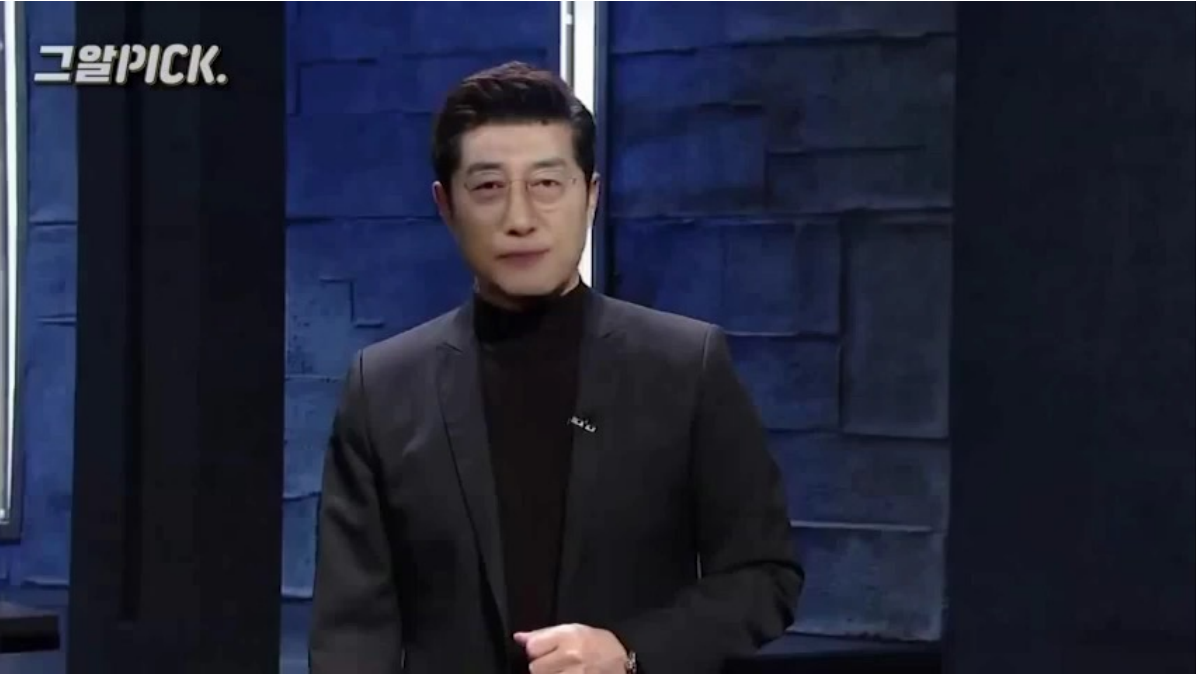


Cosine sim: 0.9358 | 김종국 → 성시경



Cosine sim: 0.9437 | 규현 → 성시경

# Demo video



김상중 → 문재인



박근혜 → 김주하

# Conclusion

- Experience in synthesizing videos from data collection to demo video
  - 논문리뷰 → 코드실습 → 데이터셋 수집 → 모델학습 → 데모영상
- Voice
  - Sampling rate, batch size, iteration, ...
  - 정량 및 정성평가
- Video
  - Average similarity between the two videos
- Future work
  - 음성) Source speaker의 음성내용에 Target speaker 목소리 톤을 입히는 경우 Source speaker의 음성 길이에 맞춰 Target speaker의 목소리 톤을 합성하다 보니 자연스러운 음성 변환이 힘든 경우가 있다.
    - Target speaker의 corpus당 음성 길이를 유지하기 위한 loss term을 학습과정에 추가하여 Target speaker의 자연스러운 음성 변환
  - 클래스당 데이터의 개수를 늘리고 좀 더 다양한 클래스를 구성하여 모델을 학습하도록 한다
  - Voice conversion에서 feature를 추출한 방법 상 고주파수 영역의 소리, phase 정보 손실
    - MFCC로 feature 추출 방법을 변경
    - wav 파일 자체를 feature로 사용(학습 시간이 오래 걸리기 때문에 추후 예정)



# Q & A

---