2022 AI보안연구센터 인턴 미팅

Research REPORT

22년 11월 21일

20170622

이종헌

Task

Deepfake Voice 탐지의 이해 및
경량화를 통해 안드로이드 앱으로 구현

1. Deepfake Voice를
탐지하는 모델을 이해하고

2. 딥러닝 모델을 경량화하여

3. 안드로이드 앱으로 구현

정수환 [IT융합]	**★주제1: Deepfake voice 탐지의 이해 및 PyTorch를 이용한 구현**
	설명: 음성 합성 기술을 악용하는 Deepfake voice를 탐지하는 모델을 이해하고 오픈소스를 기반으로 이를 pytorch를 통해 구현해본다.
	★주제2: Deepfake voice 생성에 활용되는 TTS의 이해 및 구현
	설명: Deepfake voice 생성에 활용되는 음성 합성 (TTS: Text-to-Speech) 모델에 대해 이해하고 오픈소스를 기반으로 이를 구현해본다. TTS는 원하는 음성을 학습하여 텍스트로 원본 음성과 똑같은 목소리로 음성을 생성하는 방법.
	★주제3: Deepfake voice 생성에 활용되는 VC의 이해 및 구현
	설명: Deepfake voice 생성에 활용되는 음성 합성 (VC: Voice Conversion) 모델에 대해 이해하고 오픈소스를 기반으로 이를 구현해본다. Voice Conversion은 소스 음성을 원하는 음성과 유사하게 변조하는 방법.
	★주제4: Deepfake voice 탐지의 이해 및 경량화를 통해 안드로이드 앱으로 구현
	설명: 음성 합성 기술을 악용하는 Deepfake voice를 탐지하는 모델을 이해하고 오픈소스를 기반으로 딥러닝 모델을 경량화하여 안드로이드 앱으로 구현해 본다.
	★주제5: 안드로이드 환경의 보안/해킹 앱 개발
	설명: 안드로이드 환경에서의 보안 혹은 해킹 관련 프로그램을 구현 (공인인증서 탈취 프로그램 제작, 코드 인젝션 등 다양한 보안 혹은 해킹 프로그램)

Task- Goal

Phuc님이 만든 Deepfake Voice 탐지 모델을 안드로이드에서 구현해봐라

하나의 문장이지만
그 안에는 수많은 세부사항들로 나뉨

정수환 [IT융합]	**★주제1: Deepfake voice 탐지의 이해 및 PyTorch를 이용한 구현**
	설명: 음성 합성 기술을 악용하는 Deepfake voice를 탐지하는 모델을 이해하고 오픈소스를 기반으로 이를 pytorch를 통해 구현해본다.
	★주제2: Deepfake voice 생성에 활용되는 TTS의 이해 및 구현
	설명: Deepfake voice 생성에 활용되는 음성 합성 (TTS: Text-to-Speech) 모델에 대해 이해하고 오픈소스를 기반으로 이를 구현해본다. TTS는 원하는 음성을 학습하여 텍스트로 원본 음성과 똑같은 목소리로 음성을 생성하는 방법.
	★주제3: Deepfake voice 생성에 활용되는 VC의 이해 및 구현
	설명: Deepfake voice 생성에 활용되는 음성 합성 (VC: Voice Conversion) 모델에 대해 이해하고 오픈소스를 기반으로 이를 구현해본다. Voice Conversion은 소스 음성을 원하는 음성과 유사하게 변조하는 방법.
	★주제4: Deepfake voice 탐지의 이해 및 경량화를 통해 안드로이드 앱으로 구현
	설명: 음성 합성 기술을 악용하는 Deepfake voice를 탐지하는 모델을 이해하고 오픈소스를 기반으로 딥러닝 모델을 경량화하여 안드로이드 앱으로 구현해 본다.
	★주제5: 안드로이드 환경의 보안/해킹 앱 개발
	설명: 안드로이드 환경에서의 보안 혹은 해킹 관련 프로그램을 구현 (공인인증서 탈취 프로그램 제작, 코드 인젝션 등 다양한 보안 혹은 해킹 프로그램)

Task- Phuc's Deepfake Voice 탐지하는 모델

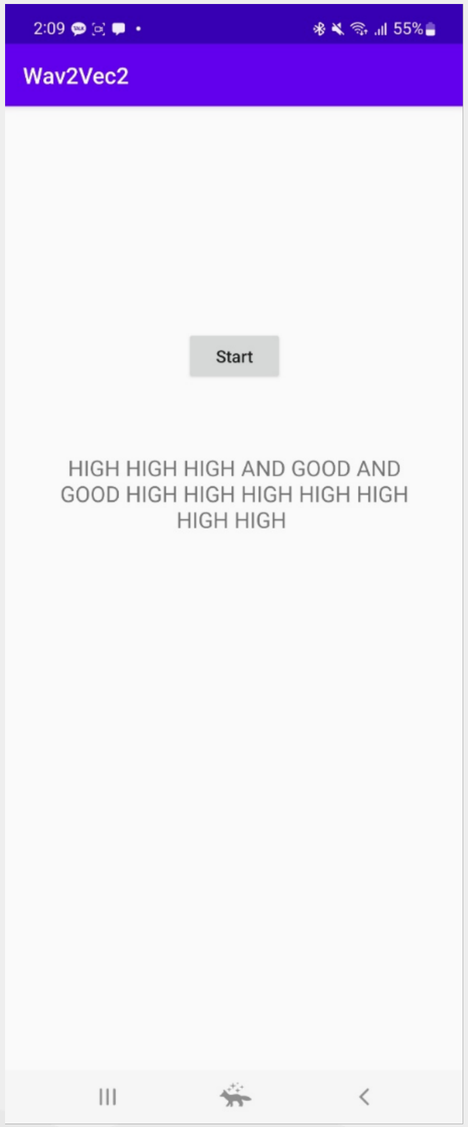
구조 -> 단순한 모델 하나로 이루어진 SW가 아님

모델의 Input : Audio
모델의 Output : Spoof or Bonafide

처음엔 간단한 모델인 줄 파악함.

예시) Android Demo App - Speech Recognition
모델의 Input : Audio
모델의 Output : STT된 Text

단순한 모델 하나
데이터 전처리가 불필요



Task- Phuc's Deepfake Voice 탐지하는 모델

구조 -> 단순한 모델 하나로 이루어진 SW가 아님

기본적인 탐지모듈의 Flow

1. wav 변수에 탐지하고 싶은 Audio를 저장

2. parse_input(wav)를 통해 Audio의 Feature를 뽑아냄

3. model 함수에 2번에서의 Feature들을 넣어서 결과값을 얻음

```
wav = "/root/dataset/Dataset/ASVspoof/LA/ASVspoof2019_LA_eval/flac/LA_E_5085671.flac"
# Extract the feature of the audio
x_inp, bio_inp, bio_length = parse_input(wav)

# use the model to calculate the prediction value
out, _ = model(x_inp, bio_inp, bio_length)
```

Task- Phuc's Deepfake Voice 탐지하는 모델 - 전처리 과정

2. parse_input(wav)를 통해 Audio의 Feature를 뽑아냄

함수 parse_input(file_path)

input : 오디오 파일의 경로

output : 오디오 파일의 Feature를 return

-> bio_inp, bio_length = get_Bio(X_pad, fs)

```
def parse_input(file_path):
    cut = 64600  # take ~4 sec audio (64600 samples)
    X, fs = librosa.load(file_path, sr=16000)
    X_pad = pad(X, cut)
    x_inp = Tensor(X_pad)
    bio_inp, bio_length = get_Bio(X_pad, fs)


    return x_inp.unsqueeze(0).to(device), bio_inp.unsqueeze(0).to(device), bio_length.to(device)
```

Task- Phuc's Deepfake Voice 탐지하는 모델 - 전처리 과정

-> bio_inp, bio_length = get_Bio(X_pad, fs)

함수 get_Bio(X_pad, fs)

input : Padded 된 Audio 데이터, Sampling Rate

output : bio_inp, bio_length return

-> bio = wav2bio(X_pad, fs)

```
def get_Bio(X_pad, fs):

    bio = wav2bio(X_pad, fs)
    # bio_length = len(bio)
    bio_inp = torch.IntTensor(bio)
    bio_length = torch.IntTensor([len(bio)])
    return bio_inp, bio_length
```

Task- Phuc's Deepfake Voice 탐지하는 모델 - 전처리 과정

-> bio = wav2bio(X_pad, fs)

wav2bio 부터는 라이브러리 단에서 선언된 함수
wav2bio함수를 사용하기 위해서는 전처리가 필요함

데이터 전처리의 전처리

```
    # Load model
models = {}
for cls in ["breath", "silence", "speech"]:
    fp = open(here + "/out/{}.gmm".format(cls), "rb")
    models[cls]=pickle.load(fp)
    fp.close()
biotype = {
    "silence":0,
    "breath":1,
    "speech":2
}
gmm_classifier = GMMClassifier(models)

silence_validator = ClassifierValidator(gmm_classifier, "silence")
speech_validator = ClassifierValidator(gmm_classifier, "speech")
breath_validator = ClassifierValidator(gmm_classifier, "breath")
# Tokennizer
analysis_window_per_second = 1. / ANALYSIS_STEP

min_seg_length = 0.2 # second, min length of an accepted audio segment
max_seg_length = 10 # seconds, max length of an accepted audio segment
max_silence = 0.3 # second, max length tolerated of tolerated continuous signal that's not from the same class

tokenizer = StreamTokenizer(validator=breath_validator, min_length=int(min_seg_length * analysis_window_per_second),
                                    max_length=int(max_seg_length * analysis_window_per_second),
                                    max_continuous_silence= max_silence * analysis_window_per_second,
                                    mode = StreamTokenizer.DROP_TRAILING_SILENCE)
```

Task- Phuc's Deepfake Voice 탐지하는 모델 - 전처리 과정

데이터 전처리의 전처리
GMM을 활용한 피쳐 엔지니어링을 해야함
3가지 biotype : breath, silence, speech

새로운 모델의 등장

GMMClassifier를 통해
모델을 파라미터로 넣어
gmm_classifier에 저장
ClassifierValidator를 통해
silence_validator
speech_validator
breath_validator
를 추출함

```
models = {}
for cls in ["breath", "silence", "speech"]:
    fp = open(here + "/out/{}.gmm".format(cls), "rb")
    models[cls]=pickle.load(fp)
    fp.close()
biotype = {
    "silence":0,
    "breath":1,
    "speech":2
}
gmm_classifier = GMMClassifier(models)

silence_validator = ClassifierValidator(gmm_classifier, "silence")
speech_validator = ClassifierValidator(gmm_classifier, "speech")
breath_validator = ClassifierValidator(gmm_classifier, "breath")
```

Task- Phuc's Deepfake Voice 탐지하는 모델 - 전처리 과정

wav2bio

함수 wav2bio(data, sr)

input : Padded 된 Audio 데이터, Sampling Rate

output : result

gmm_classifier의 predict를 통해 해당 음성파일이 silence, speech, breath 중 어디에 해당한지를 Result라는 리스트에 저장 후 리턴해 모델을 통해

```
# Load data

def wav2bio(data, sr):
    # data, sr = sf.read(wav_path)
    lfcc = VectorDataSource(data=extract_lfcc(sig=data,sr=sr),scope=15)
    # tokenized:
    lfcc.rewind()
    data = lfcc.read()
    # print(data)
    result = []
    while (data is not None):
        result.append(biotype[gmm_classifier.predict(data)[0][0]])
        data = lfcc.read()
    return result
```

Task- Phuc's Deepfake Voice 탐지하는 모델

여기까지가 2번까지의 과정

3번은 2번을 통해서 얻은 값들을 모델에 넣어서 결과를 확인하면 됨

기본적인 탐지모듈의 Flow

1. wav 변수에 탐지하고 싶은 Audio를 저장

2. parse_input(wav)를 통해 Audio의 Feature를 뽑아냄 -> 여기까지가 완료

3. model 함수에 2번에서의 Feature들을 넣어서 결과값을 얻음

```
wav = "/root/dataset/Dataset/ASVspoof/LA/ASVspoof2019_LA_eval/flac/LA_E_5085671.flac"
# Extract the feature of the audio
x_inp, bio_inp, bio_length = parse_input(wav)

# use the model to calculate the prediction value
out, _ = model(x_inp, bio_inp, bio_length)
```

Task- Phuc's Deepfake Voice 탐지하는 모델 - 정리

1. wav 변수에 탐지하고 싶은 Audio를 저장
2. parse_input(wav)를 통해 Audio의 Feature를 뽑아냄 -> 여기까지가 완료
3. model 함수에 2번에서의 Feature들을 넣어서 결과값을 얻음

-> 이 모든 과정을 Python3가 가능한 multiGPU의 워크스테이션에서 작업

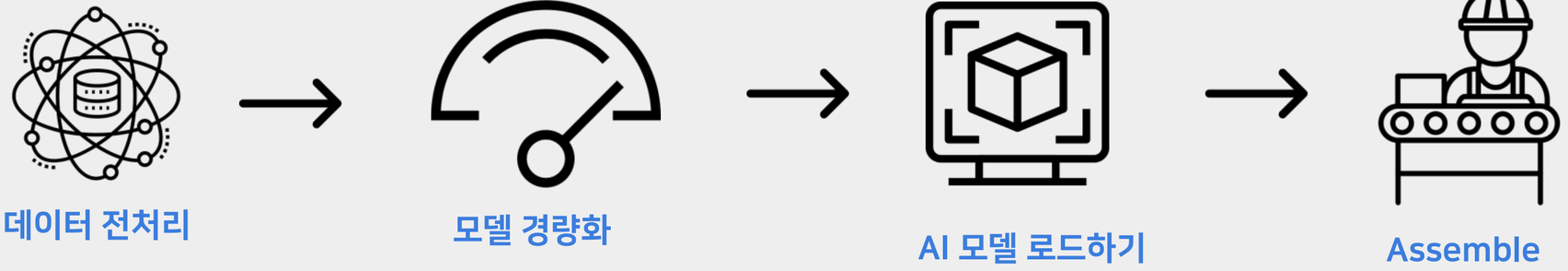
Task- 안드로이드에서 돌려봐라

-> 이 모든 과정을 Python3가 가능한 multiGPU의 워크스테이션에서 작업

이 모든 과정을 JAVA의 Android에서 구현해라

Python3

JAVA - Android



Task- Goal

Phuc님이 만든 Deepfake Voice 탐지 모델을
안드로이드에서 구현해봐라

하나의 문장이지만
그 안에는 수많은 세부사항들로 나뉨

정수환 [IT융합]	★주제1: Deepfake voice 탐지의 이해 및 PyTorch를 이용한 구현
	설명: 음성 합성 기술을 악용하는 Deepfake voice를 탐지하는 모델을 이해하고 오픈소스를 기반으로 이를 pytorch를 통해 구현해본다.
	★주제2: Deepfake voice 생성에 활용되는 TTS의 이해 및 구현
	설명: Deepfake voice 생성에 활용되는 음성 합성 (TTS: Text-to-Speech) 모델에 대해 이해하고 오픈소스를 기반으로 이를 구현해본다. TTS는 원하는 음성을 학습하여 텍스트로 원본 음성과 똑같은 목소리로 음성을 생성하는 방법.
	★주제3: Deepfake voice 생성에 활용되는 VC의 이해 및 구현
	설명: Deepfake voice 생성에 활용되는 음성 합성 (VC: Voice Conversion) 모델에 대해 이해하고 오픈소스를 기반으로 이를 구현해본다. Voice Conversion은 소스 음성을 원하는 음성과 유사하게 변조하는 방법.
	★주제4: Deepfake voice 탐지의 이해 및 경량화를 통해 안드로이드 앱으로 구현
	설명: 음성 합성 기술을 악용하는 Deepfake voice를 탐지하는 모델을 이해하고 오픈소스를 기반으로 딥러닝 모델을 경량화하여 안드로이드 앱으로 구현해 본다.
	★주제5: 안드로이드 환경의 보안/해킹 앱 개발
	설명: 안드로이드 환경에서의 보안 혹은 해킹 관련 프로그램을 구현 (공인인증서 탈취 프로그램 제작, 코드 인젝션 등 다양한 보안 혹은 해킹 프로그램)

데이터 전처리

모델 경량화

AI 모델 로드하기

Assemble

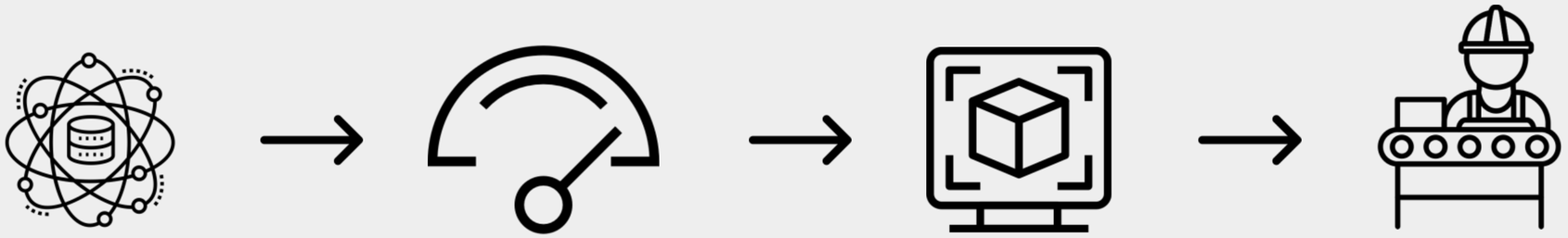
Task- This Week

홍기훈 교수님이 피드백 주신 것 처럼
주도적으로 프로젝트 파악 작업

Phuc과의 소통을 통해 Task 명확화

정수환 [IT융합]	★주제1: Deepfake voice 탐지의 이해 및 PyTorch를 이용한 구현
	설명: 음성 합성 기술을 악용하는 Deepfake voice를 탐지하는 모델을 이해하고 오픈소스를 기반으로 이를 pytorch를 통해 구현해본다.
	★주제2: Deepfake voice 생성에 활용되는 TTS의 이해 및 구현
	설명: Deepfake voice 생성에 활용되는 음성 합성 (TTS: Text-to-Speech) 모델에 대해 이해하고 오픈소스를 기반으로 이를 구현해본다. TTS는 원하는 음성을 학습하여 텍스트로 원본 음성과 똑같은 목소리로 음성을 생성하는 방법.
	★주제3: Deepfake voice 생성에 활용되는 VC의 이해 및 구현
	설명: Deepfake voice 생성에 활용되는 음성 합성 (VC: Voice Conversion) 모델에 대해 이해하고 오픈소스를 기반으로 이를 구현해본다. Voice Conversion은 소스 음성을 원하는 음성과 유사하게 변조하는 방법.
	★주제4: Deepfake voice 탐지의 이해 및 경량화를 통해 안드로이드 앱으로 구현
	설명: 음성 합성 기술을 악용하는 Deepfake voice를 탐지하는 모델을 이해하고 오픈소스를 기반으로 딥러닝 모델을 경량화하여 안드로이드 앱으로 구현해 본다.
	★주제5: 안드로이드 환경의 보안/해킹 앱 개발
	설명: 안드로이드 환경에서의 보안 혹은 해킹 관련 프로그램을 구현 (공인인증서 탈취 프로그램 제작, 코드 인젝션 등 다양한 보안 혹은 해킹 프로그램)

-> 데이터 전처리를 기존에서는 Python의 Multi GPU 환경에서 진행했는데
JAVA의 안드로이드 환경에서 진행해라



Task- This Week

-> 데이터 전처리를 기존에서는 Python의 Multi GPU 환경에서 진행했는데
JAVA의 안드로이드 환경에서 진행해라

기간 내에 전부를 다 진행하는데 있어서 시간이 부족함을 파악
JAVA와 안드로이드 모두 처음 겪어보는 환경

-> 이번주 데이터 전처리를 JAVA의 안드로이드 환경에서 시도했지만 처음 겪어보는 언어와 환경이라 JAVA 문법과 안드로이드에 대한 공부가 필요
-> 4가지 과정 모두를 진행하는 것은 기간 내에 완수하는 것을 어려워 보이고 하나의 파트를 맡아서 진행하는 것이 좋아보임