위급상황 판별

황세원
20181080 정보통계학과

정승혜
20191071 정보통계학과

CONTENTS.

01

중간 발표 요약

주제 선정, 데이터,
CNN, LSTM

주제

01. 필요성

장난전화로 인해 발생되는 문제들

02. 관련 연구

CCTV 응급상황에 따른 지능형 음성인식 시스템 구현 연구

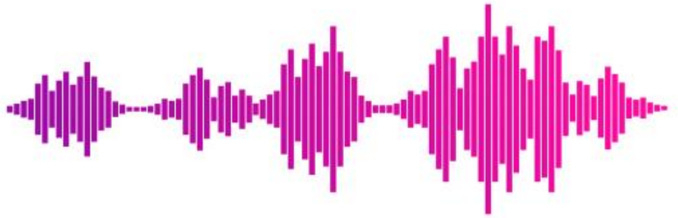
03. 독창성

음성인식을 어플리케이션을 통해 실행함으로써, 빠른 판별

데이터

01. 음성 데이터 특징

음성의 특징 - 목소리, 음향, 발음의 종류, 성별, 음색, 높이



02. 사용 데이터

AI 허브

일반 - "감정분류를 위한 음성 데이터 셋"

위급 - "위급상황 음성/음향"

- 7가지 감정(happiness, angry, disgust, fear, neutral, sadness, surprise)에 대해 5명이 라벨링

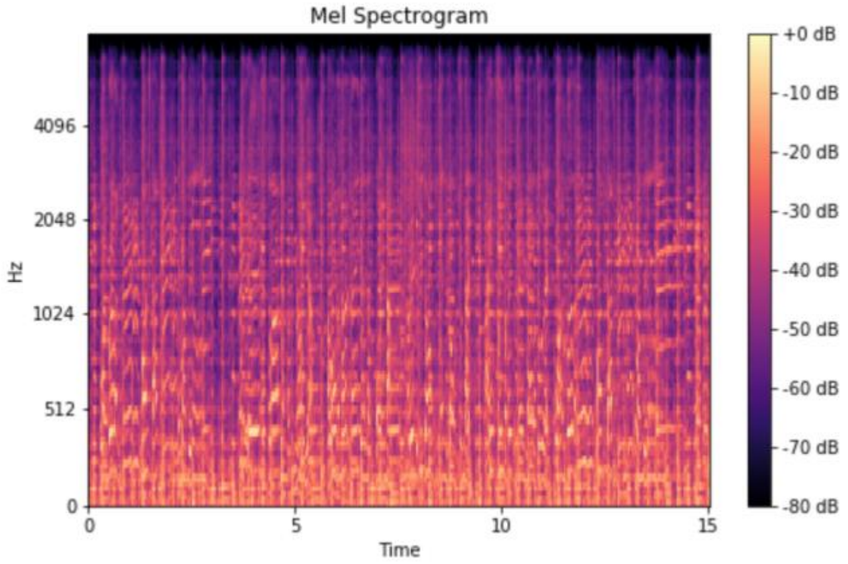
wav_id	발화문	상황	1번 감정	1번 감정세기	2번 감정	2번 감정세기	3번 감정	3번 감정세기	4번 감정	4번감정세기	5번 감정	5번 감정세기
5ee1e7939aa8ea0eec53fac7	나 어제 헤어졌어	sad	Sadness	1	Sadness	1	Sadness	1	Sadness	1	Sadness	1
5ee1e7a379bf120ed2b81ba2	어쩌다 보니까 그렇게 됐네	sad	Sadness	1	Sadness	1	Neutral	0	Sadness	1	Sadness	1
5ed9793b9aa8ea0eec53f7f1	유기견 다큐멘터리를 봤는데 무책임한 사람들 때문에 너무 화가 나	disgust	Sadness	2	Sadness	2	Angry	1	Angry	1	Angry	1
5ed979582880d70f286128c3	버려지는 유기견들의 생활을 다른 다큐멘터리 얻어	disgust	Sadness	1	Sadness	2	Sadness	1	Angry	1	Sadness	2
5ed9797b1dcf350eeded50b8	작년보다 5배는 더 늘어나는 것 같대 최근 들어 더 늘어나고 있고	disgust	Fear	1	Sadness	1	Sadness	1	Angry	1	Sadness	2
5ed9799ac90a530ee56b5f6b	정부보조금이랑 사람들의 기부금으로 운영되고 있어	disgust	Neutral	0	Neutral	0	Sadness	1	Sadness	1	Sadness	2
5ed979dc7e21a10eee253e90	맞아 벌어지는 대부분의 아이들이 다 큰 강아지 들였어 아직도 상처	disgust	Sadness	2	Sadness	2	Sadness	2	Sadness	2	Sadness	2
5ed97a22c90a530ee56b5f6c	어제저녁에 진짜 무서웠어	fear	Fear	2	Fear	2	Fear	1	Fear	1	Fear	2
5ed97a381dcf350eeded50bc	친구랑 약속 있어 나갔는데 밥 먹고 이따가 지진이 발생하는 거야 엘	fear	Fear	2	Fear	2	Fear	2	Fear	1	Fear	1
5ed97a5cc90a530ee56b5f6d	큰 지진은 지나갔는데 여진이 조금씩 있는 거 같기도 해	fear	Fear	2	Fear	2	Fear	1	Fear	1	Fear	1
5ed97a779aa8ea0eec53f7f5	다른 테이블에서 밥 먹던 사람들이 도망치다가 넘어지고 그 와중에 니	fear	Sadness	1	Fear	2	Sadness	1	Fear	1	Fear	1
5ed97a977e21a10eee253e92	그렇지 않아도 오늘 친구 데리고 병원 가기로 했어	fear	Neutral	0	Sadness	1	Sadness	1	Sadness	1	Fear	2
5ed97ac29aa8ea0eec53f7f8	룸메이트와 너무 자주 싸우게 돼	anger	Disgust	2	Angry	2	Angry	1	Angry	1	Fear	1

위급상황 분야	분류체계	구축규모 (시간)	비율(%)
치안안전	강제추행	100	2.9
	강도범죄	100	2.9
	절도범죄	100	2.9
	폭력범죄	200	5.7
소방안전	갇힘	100	2.9
	전기사고	200	5.7
	가스사고	150	4.3
	화재	450	12.9
	응급의료	450	12.9
자연재해	태풍/강풍	300	8.6
	지진	100	2.9
사고발생	낙상	150	4.3
	붕괴사고	100	2.9
일반(위급)	도움요청	300	8.6
일반(정상)	실내	500	14.3
	실외	200	5.7
합계		3,500	100%

CNN

melspectrogram

melspectrogram이란, 인간이 이해하기 힘든 Spectrogram의 y축을 Mel Scale로 변환한 것.

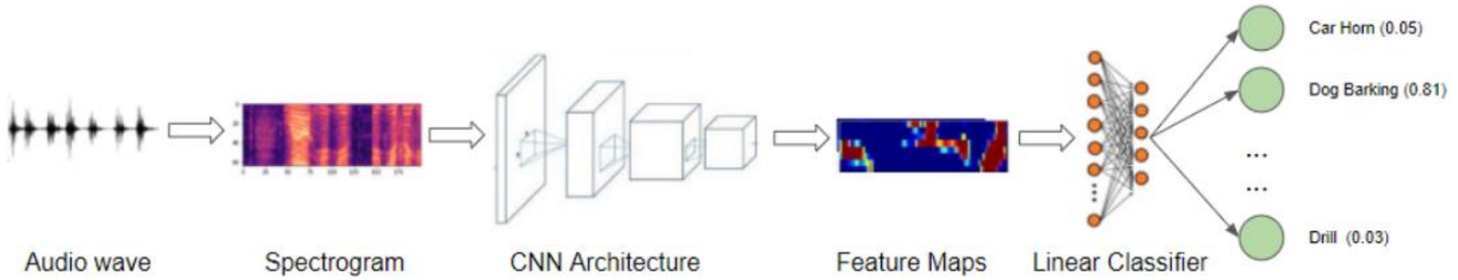


* mel filter bank: 사람이 인식할 수 있는 주파수 영역으로 mel scale 변환이 이뤄지는 구간

*spectrogram: 소리나 파동을 시각화하여 파악하기 위한 도구로, 파형과 스펙트럼의 특징이 조합. x축은 시간축, y축은 주파수

CNN 음성분류 적용

1. 음성 -> 이미지 변환
2. 이미지 특징 추출
3. 이진 분류



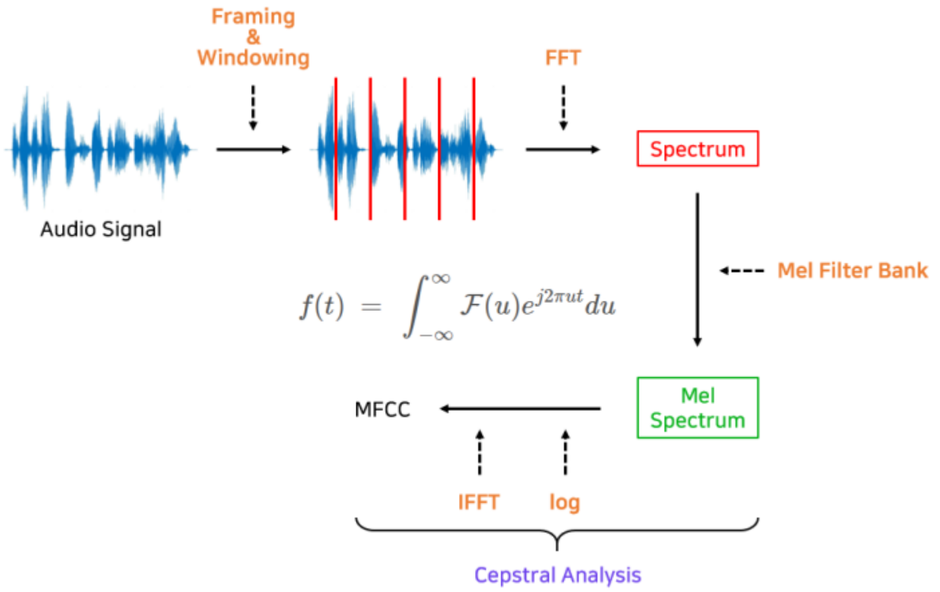
결과

train loss : 0.14
train acc: 0.95
test loss : 0.13
test acc: 0.95

LSTM

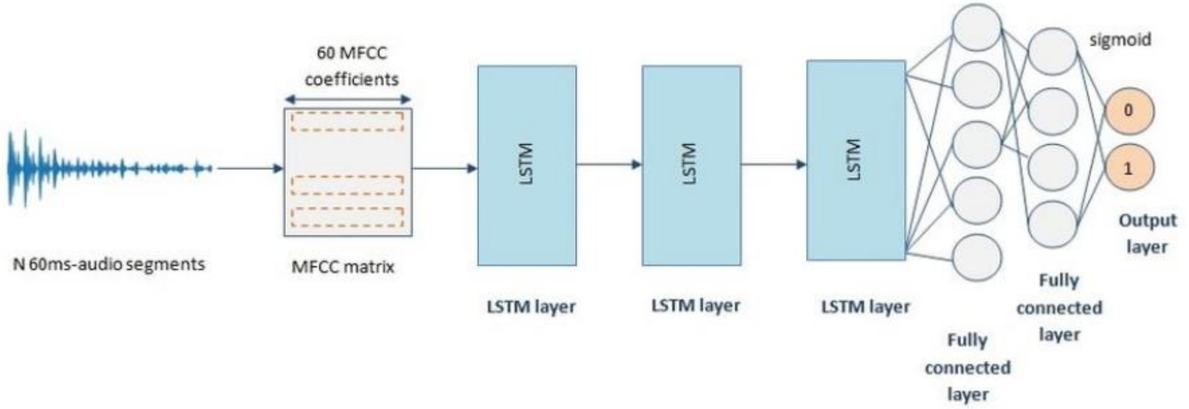
mfcc : Mel-Frequency Cepstral Coefficient

MFCC는 '음성데이터'를 '특징벡터' (Feature) 화 해주는 알고리즘



LSTM 음성분류 적용

1. 음성에서 특징 벡터 추출
2. 이진 분류



결과

train loss : 0.05
train acc: 0.98
test loss : 0.27
test acc: 0.92

02

전이학습

VGG-16

전이 학습: VGG-16

사용이유

수많은, 수천개의 데이터로 미리 학습된 이미지 데이터에서 높은 성능을 보이는 모델 VGG-16을 이용해 음성 데이터에도 성능이 좋은지 의문

전처리

위험데이터 : 1400개
일반데이터 : 1500개
5초이하로 불러온 음성 -> melspectrogram -> 이미지로 저장

모델

vgg-16의 입력층, 컨볼루션, 맥스풀링으로 구성된 19층 모델
+
dense 3층

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 150, 150, 3)]	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1180160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2359808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2359808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2359808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0

Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 4, 4, 512)	14714688
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 50)	409650
dense_1 (Dense)	(None, 20)	1020
dense_2 (Dense)	(None, 2)	42

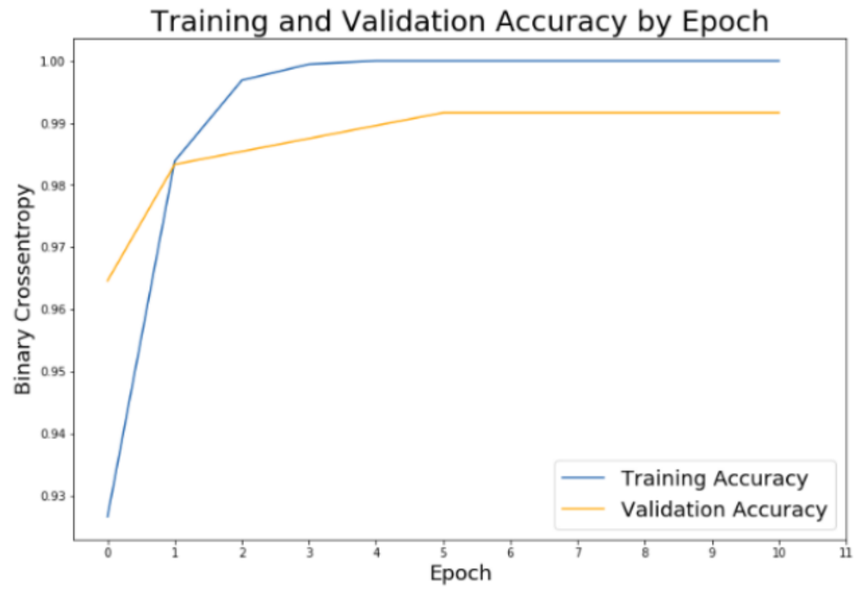
Total params: 15,125,400
Trainable params: 410,712
Non-trainable params: 14,714,688

es = EarlyStopping(monitor='val_accuracy', mode='max', patience=5, restore_best_weights=True)

```
Epoch 1/50
60/60 [==============================] - 329s 5s/step - loss: 0.4049 - accuracy: 0.9266 - val_loss: 0.0890 - val_accuracy: 0.9646
Epoch 2/50
60/60 [==============================] - 335s 6s/step - loss: 0.0545 - accuracy: 0.9839 - val_loss: 0.0792 - val_accuracy: 0.9833
Epoch 3/50
60/60 [==============================] - 335s 6s/step - loss: 0.0082 - accuracy: 0.9969 - val_loss: 0.0685 - val_accuracy: 0.9854
Epoch 4/50
60/60 [==============================] - 321s 5s/step - loss: 0.0022 - accuracy: 0.9995 - val_loss: 0.0464 - val_accuracy: 0.9875
Epoch 5/50
60/60 [==============================] - 287s 5s/step - loss: 2.0108e-04 - accuracy: 1.0000 - val_loss: 0.0462 - val_accuracy: 0.9896
Epoch 6/50
60/60 [==============================] - 288s 5s/step - loss: 1.5336e-04 - accuracy: 1.0000 - val_loss: 0.0459 - val_accuracy: 0.9917
Epoch 7/50
60/60 [==============================] - 295s 5s/step - loss: 1.2867e-04 - accuracy: 1.0000 - val_loss: 0.0453 - val_accuracy: 0.9917
Epoch 8/50
60/60 [==============================] - 317s 5s/step - loss: 1.1178e-04 - accuracy: 1.0000 - val_loss: 0.0451 - val_accuracy: 0.9917
Epoch 9/50
60/60 [==============================] - 312s 5s/step - loss: 9.8928e-05 - accuracy: 1.0000 - val_loss: 0.0445 - val_accuracy: 0.9917
Epoch 10/50
60/60 [==============================] - 289s 5s/step - loss: 8.7754e-05 - accuracy: 1.0000 - val_loss: 0.0444 - val_accuracy: 0.9917
Epoch 11/50
60/60 [==============================] - 295s 5s/step - loss: 8.0276e-05 - accuracy: 1.0000 - val_loss: 0.0440 - val_accuracy: 0.9917
```

결과

train loss: 8.0276e-05
train accuracy: 1.0000
test loss: 0.0440
test accuracy: 0.9917



01_중간 발표 요약

04_결과

02_전이학습

05_어려운 점

03_멀티모달

03

멀티모달

CNN+LSTM

멀티모달: CNN+LSTM

사용이유

발화 내용도 활용하고 싶어서 텍스트 분류를 위한 LSTM 모델과
앞에서 사용했던 CNN 모델을 합친 멀티모달 모델을 구현

전처리

위험데이터 : 1400개
일반데이터 : 1500개
7초로 통일
* LSTM : 토큰화, 불용어 제거, Okt 사용. 패딩 12
* CNN : melspectrogram

LSTM 전처리

```
Model: "model"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================================================
conv2d_input (InputLayer)    [(None, 128, 300, 1       0         []
                             )]

conv2d (Conv2D)              (None, 124, 296, 24       624       ['conv2d_input[0][0]']
                             )

max_pooling2d (MaxPooling2D)  (None, 31, 148, 24)      0         ['conv2d[0][0]']

activation (Activation)      (None, 31, 148, 24)       0         ['max_pooling2d[0][0]']

conv2d_1 (Conv2D)            (None, 31, 148, 48)       28848     ['activation[0][0]']

max_pooling2d_1 (MaxPooling2D)  (None, 7, 74, 48)      0         ['conv2d_1[0][0]']

activation_1 (Activation)    (None, 7, 74, 48)         0         ['max_pooling2d_1[0][0]']

conv2d_2 (Conv2D)            (None, 7, 74, 48)         57648     ['activation_1[0][0]']

activation_2 (Activation)    (None, 7, 74, 48)         0         ['conv2d_2[0][0]']

flatten (Flatten)            (None, 24864)             0         ['activation_2[0][0]']

dropout (Dropout)            (None, 24864)             0         ['flatten[0][0]']

embedding_input (InputLayer)  [(None, None)]           0         []

dense (Dense)                (None, 64)                1591360   ['dropout[0][0]']

embedding (Embedding)        (None, None, 100)         90200     ['embedding_input[0][0]']

activation_3 (Activation)    (None, 64)                0         ['dense[0][0]']

lstm (LSTM)                  (None, 128)               117248    ['embedding[0][0]']

dropout_1 (Dropout)          (None, 64)                0         ['activation_3[0][0]']

concatenate (Concatenate)    (None, 192)               0         ['lstm[0][0]',
                                                                   'dropout_1[0][0]']

dense_1 (Dense)              (None, 4)                 772       ['concatenate[0][0]']

dense_2 (Dense)              (None, 2)                 10        ['dense_1[0][0]']

dense_3 (Dense)              (None, 1)                 3         ['dense_2[0][0]']

=================================================================================================
Total params: 1,886,713
Trainable params: 1,886,713
Non-trainable params: 0
```

모델

CNN : 앞의 모델과 동일하게 사용
LSTM : 임베딩층, LSTM층
+ concatenate 후 dense(4) + dense (2, softmax)

결과

train loss : 0.0076
train acc: 0.9995
test loss : 0.0261
test acc: 0.9920

```
Epoch 1/20
32/32 [==============================] - 20s 126ms/step - loss: 0.7163 - accuracy: 0.5172 - val_loss: 0.6798 - val_accuracy: 0.5172
Epoch 2/20
32/32 [==============================] - 2s 57ms/step - loss: 0.6848 - accuracy: 0.5172 - val_loss: 0.6595 - val_accuracy: 0.5172
Epoch 3/20
32/32 [==============================] - 2s 55ms/step - loss: 0.6856 - accuracy: 0.5172 - val_loss: 0.6466 - val_accuracy: 0.5172
Epoch 4/20
32/32 [==============================] - 2s 57ms/step - loss: 0.6372 - accuracy: 0.5591 - val_loss: 0.6190 - val_accuracy: 0.5908
Epoch 5/20
32/32 [==============================] - 2s 57ms/step - loss: 0.5159 - accuracy: 0.7862 - val_loss: 0.3973 - val_accuracy: 0.9460
Epoch 6/20
32/32 [==============================] - 2s 57ms/step - loss: 0.3670 - accuracy: 0.9507 - val_loss: 0.3458 - val_accuracy: 0.9540
Epoch 7/20
32/32 [==============================] - 2s 57ms/step - loss: 0.2934 - accuracy: 0.9680 - val_loss: 0.2590 - val_accuracy: 0.9736
Epoch 8/20
32/32 [==============================] - 2s 57ms/step - loss: 0.2238 - accuracy: 0.9803 - val_loss: 0.2064 - val_accuracy: 0.9828
Epoch 9/20
32/32 [==============================] - 2s 57ms/step - loss: 0.1647 - accuracy: 0.9916 - val_loss: 0.1471 - val_accuracy: 0.9931
Epoch 10/20
32/32 [==============================] - 2s 57ms/step - loss: 0.1189 - accuracy: 0.9941 - val_loss: 0.1411 - val_accuracy: 0.9782
Epoch 11/20
32/32 [==============================] - 2s 56ms/step - loss: 0.0822 - accuracy: 0.9975 - val_loss: 0.0743 - val_accuracy: 0.9943
Epoch 12/20
32/32 [==============================] - 2s 57ms/step - loss: 0.0491 - accuracy: 0.9990 - val_loss: 0.0486 - val_accuracy: 0.9954
Epoch 13/20
32/32 [==============================] - 2s 57ms/step - loss: 0.0245 - accuracy: 0.9985 - val_loss: 0.0243 - val_accuracy: 0.9954
Epoch 14/20
32/32 [==============================] - 2s 58ms/step - loss: 0.0044 - accuracy: 0.9995 - val_loss: 0.0252 - val_accuracy: 0.9931
Epoch 15/20
32/32 [==============================] - 2s 57ms/step - loss: 0.0080 - accuracy: 0.9995 - val_loss: 0.0256 - val_accuracy: 0.9943
Epoch 16/20
32/32 [==============================] - 2s 57ms/step - loss: 0.0077 - accuracy: 0.9995 - val_loss: 0.0255 - val_accuracy: 0.9943
Epoch 17/20
32/32 [==============================] - 2s 57ms/step - loss: 0.0077 - accuracy: 0.9995 - val_loss: 0.0254 - val_accuracy: 0.9943
Epoch 18/20
32/32 [==============================] - 2s 56ms/step - loss: 0.0077 - accuracy: 0.9995 - val_loss: 0.0261 - val_accuracy: 0.9954
Epoch 19/20
32/32 [==============================] - 2s 57ms/step - loss: 0.0076 - accuracy: 0.9995 - val_loss: 0.0260 - val_accuracy: 0.9943
Epoch 20/20
32/32 [==============================] - 2s 57ms/step - loss: 0.0076 - accuracy: 0.9995 - val_loss: 0.0261 - val_accuracy: 0.9920
```

01_중간 발표 요약

04_결과

02_전이학습

05_어려운 점

03_멀티모달

04

결과

모델 비교

모델 비교

1. CNN

결과
train : loss: 0.14, accuracy: 0.95
test : loss: 0.13, accuracy: 0.95

장점 / 단점
이미지로 음성의 패턴을 학습 / 이미지 데이터라 전처리, 학습 속도 느림

2. LSTM

결과
train : loss: 0.05, accuracy: 0.98
test : loss: 0.27, accuracy: 0.92

장점 / 단점
음성 데이터의 시간성을 포함, 학습 속도 빠름 / 단순한 구조

3. VGG-16 전이학습

결과
train: loss: 3.2773e-05, accuracy: 1.00
test: loss: 0.03, accuracy: 0.98

장점 / 단점
고성능 모델로 정확도 높음/ 모델 구조가 기존 모델과 차이가 크게 없음

4. CNN + LSTM 멀티모달

결과
train : loss: 0.0076, accuracy: 0.9995
test : loss: 0.0261, accuracy: 0.9920

장점 / 단점
이미지, 텍스트를 두가지 고려 / 구현의 어려움

최종 모델

4. CNN + LSTM 멀티모달

결과
train : loss: 0.0076, accuracy: 0.9995
test : loss: 0.0261, accuracy: 0.99201

장점 / 단점
이미지, 텍스트를 두가지 고려 / 구현의 어려움

01_중간 발표 요약

04_결과

02_전이학습

05_어려운 점

03_멀티모달

05

어려운 점

프로젝트 진행 중
어려운 점

어려운 점

프로젝트 진행 중 어려운 점

새로운 영역

학부에서 배우지 못한 지식으로, 자료 조사, 전처리 등 새로운 영역 다수

데이터 형식

다양한 모델 이용으로, 각 모델에 맞게 데이터를 수정하는데 시간 소요

떠나간 모델들..

CNN+LSTM

연결 부분 에러 해결 실패

전이학습 : YAMnet

웃음, 짖음 또는 사이렌과 같은 521개 클래스의 오디오 이벤트 예측가능
사전 훈련 심층 신경망 YAMNet
같은 오디오 + 클래스 중 인간 소리(아기 울므, 비명, 대화 등) 구별 가능
이유를 알 수 없으나 정확도가 0.5

멀티모달 : VGG-16 + LSTM

vgg-16과 lstm 따로는 성능이 좋아
합쳐서 사용하면 더 좋은 시너지 예상하고 했으나,
역시 정확도 0.5. 해결 실패

출처, 참고

논문

조영임(Young Im Cho),and 장성순(Sung Soon Jang). "CCTV 응급상황에 따른 지능형 음성인식 시스템 구현." 한국지능시스템학회논문지 19.3 (2009): 415-420.
윤상혁 (Sanghyeuk Yoon), 전다윤 (Dayun Jeon),and 박능수 (Neungsoo Park). "CNN - LSTM 모델 기반 음성 감정인식." 한국정보처리학회 학술대회논문집 28.2 (2021): 939-941.

코드 참고

yamnet 전이학습: https://www.tensorflow.org/tutorials/audio/transfer_learning_audio
vggnet 전이학습: https://towardsdatascience.com/transfer-learning-with-vgg16-and-keras-50ea161580b4
cnn+lstm 멀티모달 : https://www.kaggle.com/code/nikhilroxtomar/embeddings-cnn-lstm-models-lb-0-683
CNN: https://wiserloner.tistory.com/
LSTM: https://wikidocs.net/64076

데이터

"위급상황 음성/음향", 아이엠알 (https://aihub.or.kr/aidata/30742)
"감정 분류를 위한 대화 음성 데이터셋", KAIST 인공지능연구소 (https://aihub.or.kr/opendata/keti-data/recognition-laguage/KETI-02-002)

출처, 참고

이미지

https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/
https://www.sciencedirect.com/science/article/abs/pii/S1746809421007047#!
https://www.youtube.com/watch?v=xUZNkmZm5ps
https://www.youtube.com/watch?v=fSPB3bjo63c
https://www.shutterstock.com/ko/image-vector/pink-violet-gradient-equalizer-isolated-on-1198953460
https://librosa.org/doc/latest/index.html
https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5
https://brightwon.tistory.com/11
https://hyongdoc.tistory.com/403
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.urbanbrush.net%2Fdownloads%2F%25EA%25B3%25A0%25EB%25AF%25BC-%25EC%259D%25BC%25EB%259F%25AC%25EC%258A%25A4%25ED%258A%25B8-ai-%25EB%258B%25A4%25EC%259A%25B4%25EB%25A1%259C%25EB%2593%259C-download-suffering-vector%2F&psig=AOvVaw1f6KcK6s9fRDGit2Qfdqas&ust=1655283173777000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCPCzpsblrPgCFQAAAAAdAAAAABAK
https://www.sciencedirect.com/science/article/abs/pii/S1746809421007047#!
https://www.shutterstock.com/ko/image-vector/pink-violet-gradient-equalizer-isolated-on-1198953460
https://librosa.org/doc/latest/index.html
https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5


THANK YOU!
DO YOU HAVE ANY
QUESTION?
YES
NO