

SW중심대학 디지털 경진대회

AI 부문

이세계·아이돌

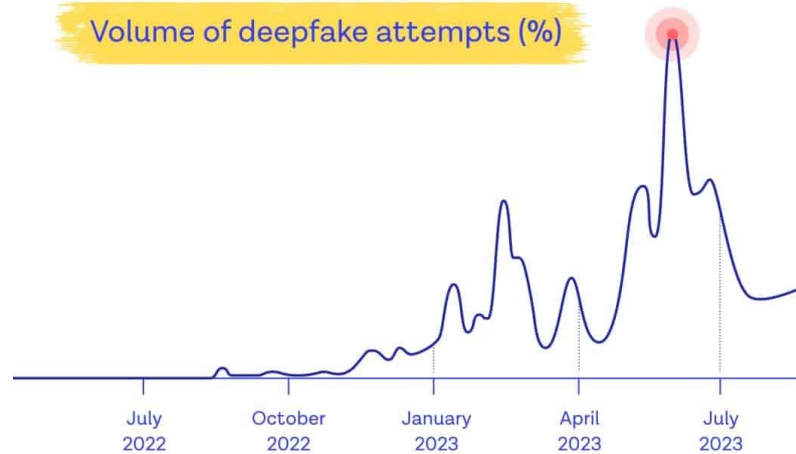
김준영, 김정우, 박상혁

Index

1. Object Setting
2. Data Inspection
3. Data Preprocessing
4. Modeling
5. Evaluation of the Results
6. Deployment

Object Setting

Volume of deepfake attempts (%)



onfido Identity Fraud Report 2024

<https://onfido.com/landing/identity-fraud-report/>

시간이 지남에 따라 deepfake의 시도가 증가함을 알 수 있습니다.

해당 통계를 통해 시간이 지남에 따라 deepfake fraud의 시도 또한 증가할 것임을 유추할 수 있습니다.

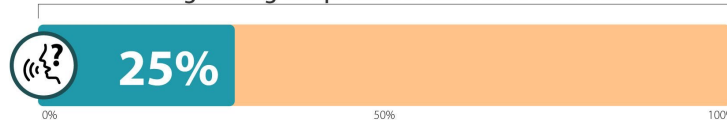
25%가 deepfake audio와 bona-fide audio를 구분하지 못한다고 답하였습니다.

이를 통해 deepfake audio를 구별할 수 있는 AI 모델의 도입이 시급하다는 것을 알 수 있습니다.



Difficulty in Identifying Deepfake Audio

Percentage of people who have difficulty distinguishing deepfake voices from real voices.



Source: LocalCircles (2023)

<https://keepnetlabs.com/blog/deepfake-statistics-and-trends-about-cyber-threats-2024>



Public Unawareness and Misconception

People surveyed around the world.



Source: Iproov

keepnet

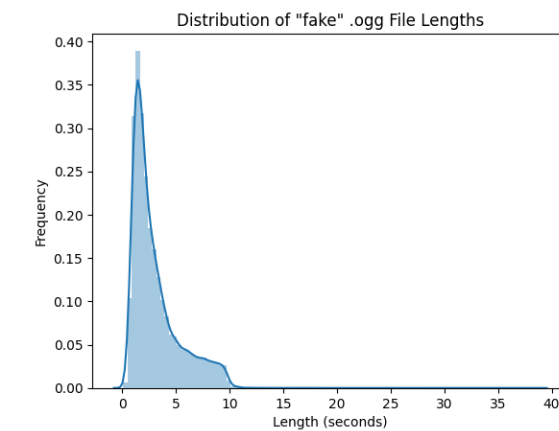
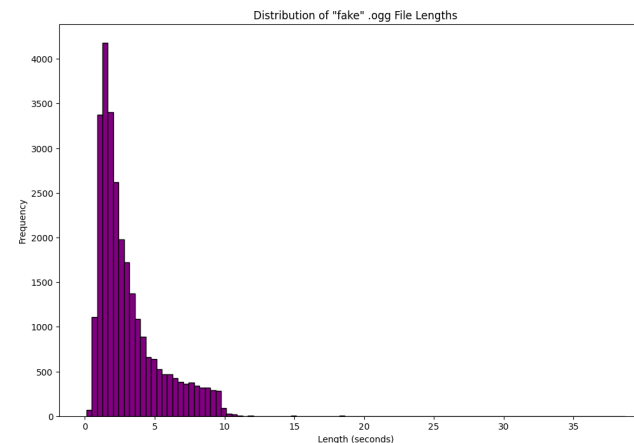
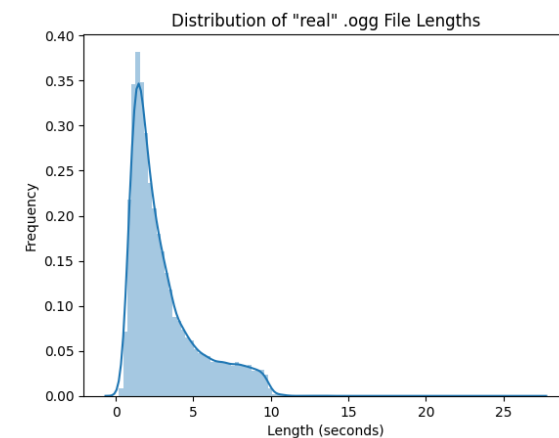
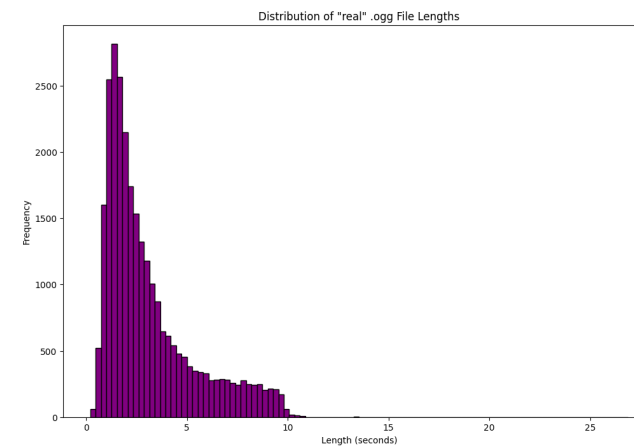
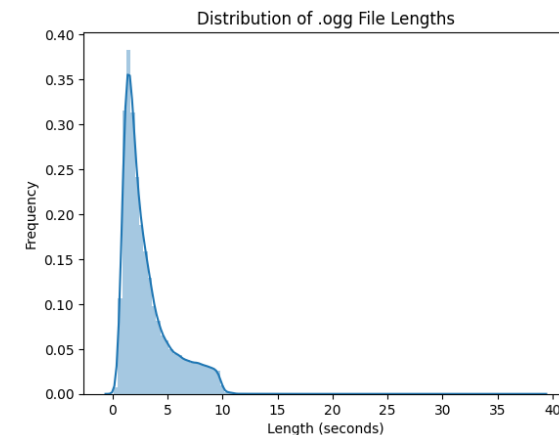
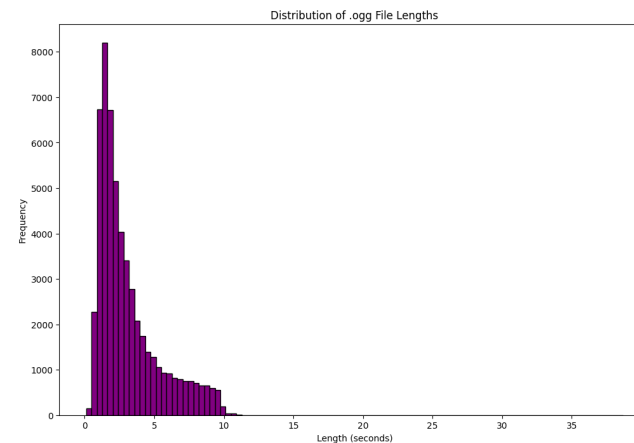
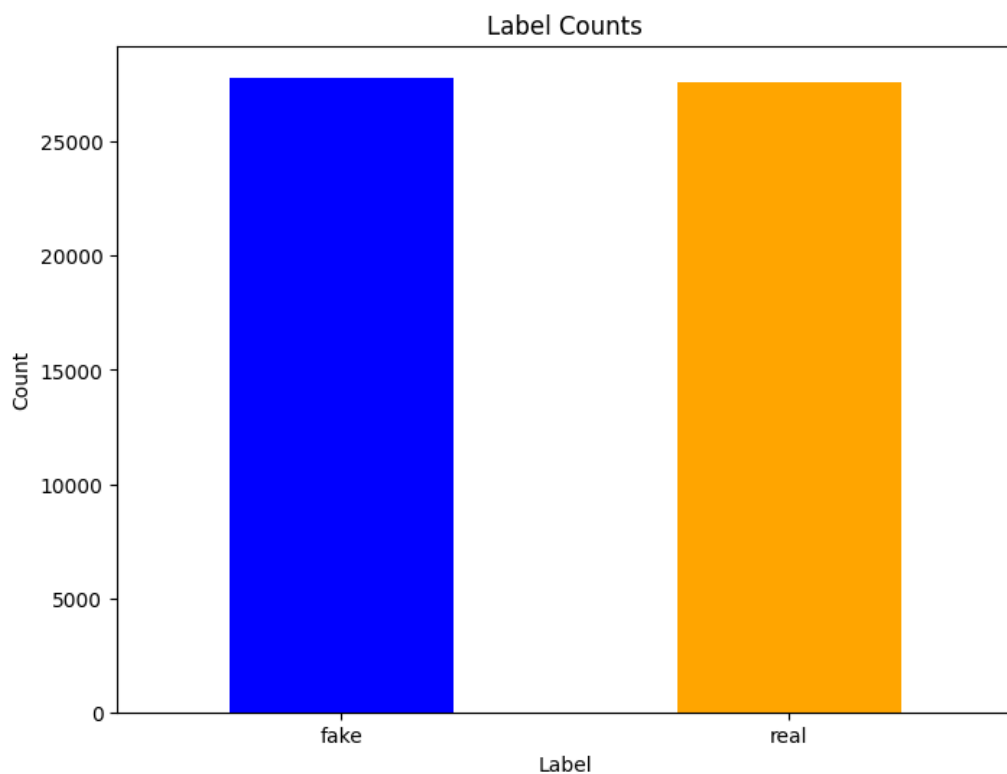
<https://keepnetlabs.com/blog/deepfake-statistics-and-trends-about-cyber-threats-2024>

사람들의 71%가 deepfake가 무엇인지 알지 못한다고 답하였습니다.

이를 통해 대부분의 사람들이 deepfake fraud에 속거나 당할 것이라고 예상할 수 있습니다.

Data Inspection

불균형 x
real인 데이터는 audio시간이 조금 길지만
큰 상관관계가 있다고 하기는 애매함.



• **train [폴더]**

- 55438개의 학습 가능한 32kHz 로 샘플링 된 오디오(ogg) 샘플
- 방음 환경에서 녹음된 진짜 사람 목소리(Real) 샘플과 방음 환경을 가정한 가짜 생성 목소리(Fake) 샘플
- 각 샘플 당 한명의 진짜 혹은 가짜 목소리가 존재

• **test [폴더]**

- 50000개의 [5초 분량](#)의 32kHz 로 샘플링 된 평가용 오디오(ogg) 샘플
- TEST_00000.ogg ~ TEST_49999.png
- 방음 환경 혹은 방음 환경이 아닌 환경 모두 존재하며, 각 샘플 당 최대 2명의 목소리(진짜 혹은 가짜)가 존재

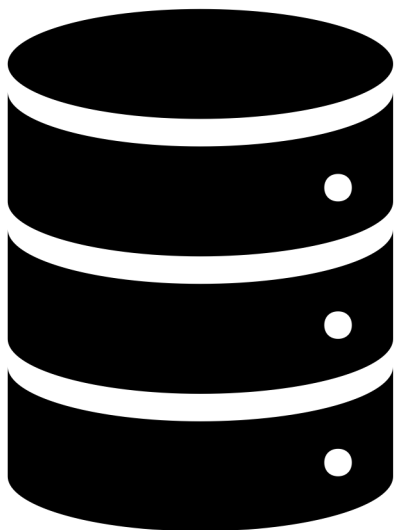
• **unlabeled_data [파일]**

- 1264개의 [5초 분량](#)의 학습 가능한 32kHz 로 샘플링 된 Unlabeled 오디오(ogg) 샘플
- 평가용 오디오(ogg) 샘플과 동일한 환경이지만 Label은 제공되지 않음

Data Inspection

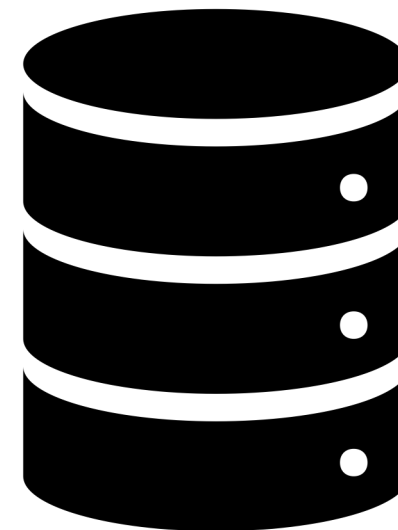
distribution of train and test data is different

train



1. train, unlabeled data를 활용해 최대한 test data의 분포에 맞춰준다.
2. test데이터의 분포를 train data의 분포에 맞게 변형시킨다.

test, unlabeled



32kHz

only one person(real or fake)

soundproof(방음) environment

32kHz



up to 2 person(real or fake)

both soundproof(방음) environment and not






Data Inspection

train, unlabeled data를 활용해 최대한 test data의 분포에 맞춰준다.

1. 2개의 음성 데이터를 섞는다. 
2. 노이즈를 추가한다. 
3. 외부 소음을 추가한다. (외부 데이터 사용 불가로 불가능)

test데이터의 분포를 train data의 분포에 맞게 변형시킨다.

1. 2명의 화자가 있는 경우 분리를 한다. 
2. 외부 소음을 제거한다. 
3. 오디오에 화자가 몇 명 있는지(0, 1, 2) 각각 분류한다. 

주어진 데이터 셋에서 효과적임


Data Inspection

train

person	fake	real
1	1	0
1	0	1

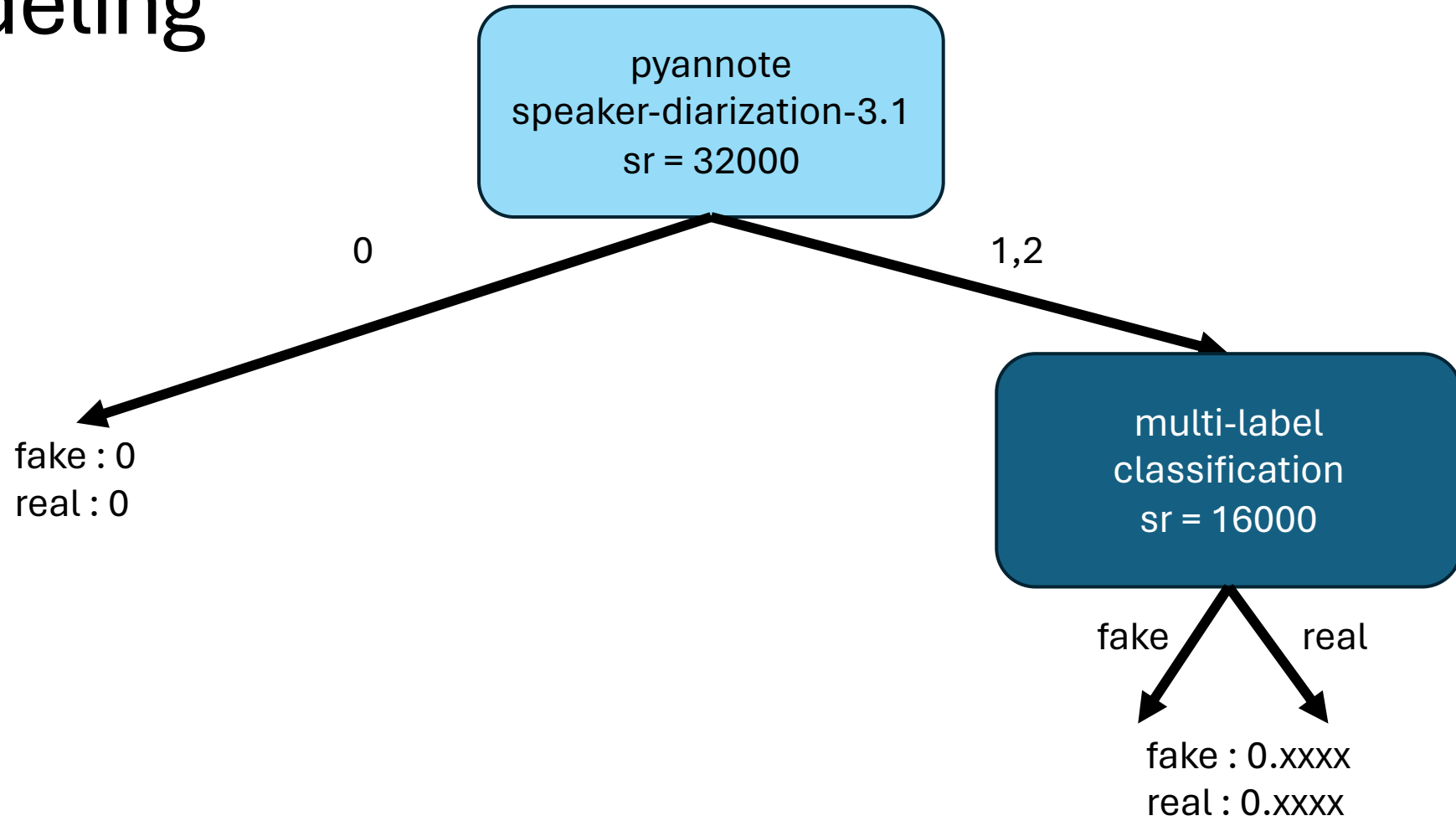
speaker diarization

test



person	fake	real
0	0	0
1	1	0
1	0	1
2	1	0
2	0	1
2	1	1

Modeling

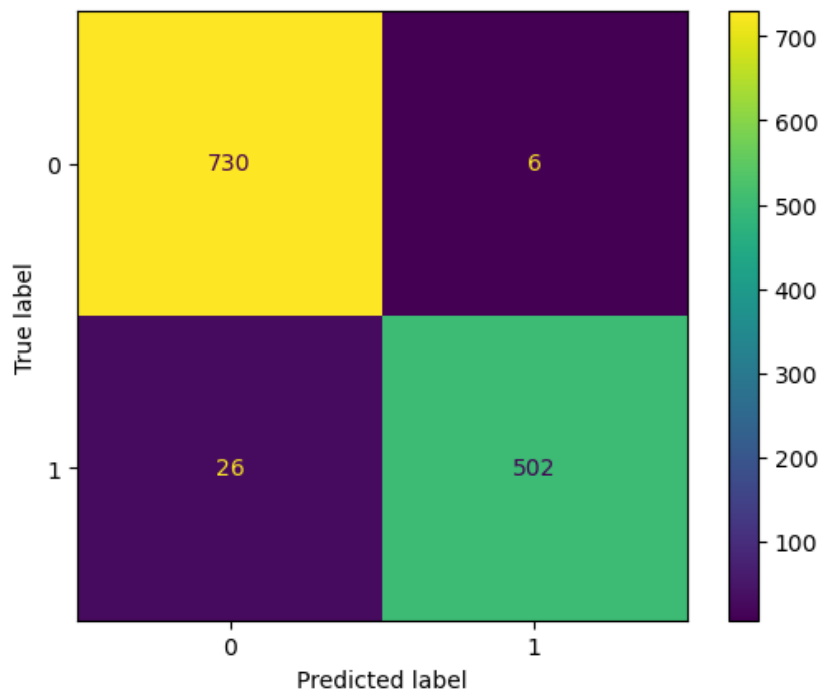


pyannote/speaker-diarization-3.1

검증 결과

accuracy : 0.9746835443037974

f1-score : 0.9785522788203753



	id	count
0	ABJGMLHQ	2
1	ABKEEJML	2
2	ABPTCSCT	2
3	ABWKLTSV	1
4	ABXKIWRT	1
...
1259	ZXXYQAZT	1
1260	ZXYKTQGY	1
1261	ZYDCWUVX	2
1262	ZYOASADS	2
1263	ZYUAIDJD	2

모델 검증을 위해 1264개의
unlabeled data를 라벨링

Data preprocessing

1. 일반적인 음성 데이터의 sampling rate가 16kHz미만임을 고려할 때, **실용적인 측면에서 32kHz대신 16kHz의 데이터를 사용**하는 것이 좋을 것이라고 판단해 모든 데이터를 16kHz로 변환해 사용
2. 5초 미만의 데이터의 경우 **동일한 음성이 반복**되도록, 5초 이상의 데이터의 경우 **앞부분의 5초만 사용**(padding).
3. padding처리 된 음성 데이터 2개를 더하고 각각의 데이터를 통해 label을 다시 만들어 줌.

Data preprocessing

```
class Dataset_ASVspoof_train(Dataset):
    def __init__(self, args, list_IDs, labels, algo):
        self.list_IDs = list_IDs
        self.labels = labels
        self.algo = algo
        self.args = args
        self.cut = 80000

    def get_labels(self):
        return self.labels

    def __len__(self):
        return len(self.list_IDs)

    def __getitem__(self, index):
        X, fs = librosa.load('./train16000/'+self.list_IDs[index]+'wav', sr=16000)

        Y1 = pad(X, self.cut)
        tmp = np.random.randint(0, len(self.list_IDs))
        X, fs = librosa.load('./train16000/'+self.list_IDs[tmp]+'wav', sr=16000)
        Y2 = pad(X, self.cut)
        Y1 = process_Rawboost_feature(Y1 + Y2, fs, self.args, self.algo)
        return Tensor(Y1), Tensor([(self.labels[index] and self.labels[tmp])^1, self.labels[index] or self.labels[tmp]])
```

```
class ForeverDataIterator:
    r"""A data iterator that will never stop producing data"""

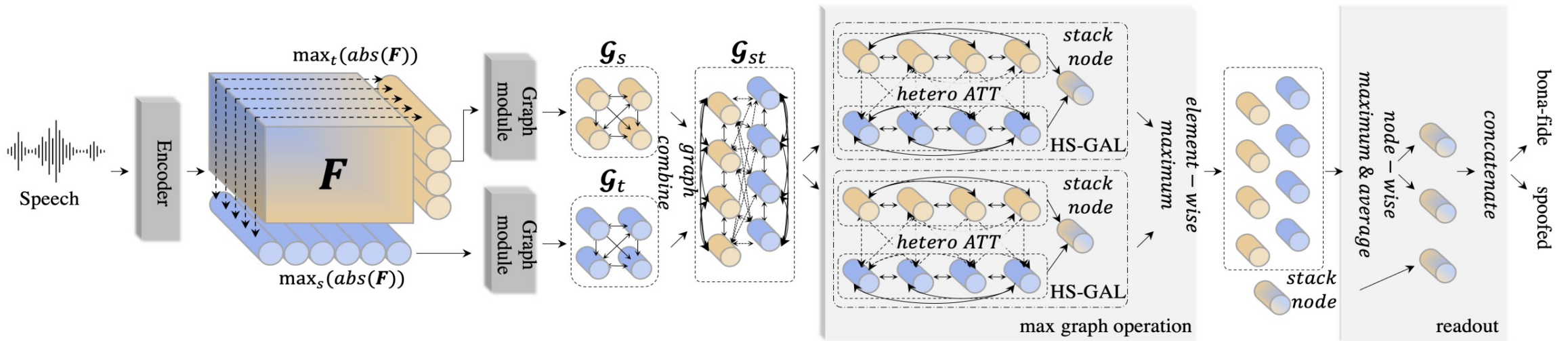
    def __init__(self, data_loader: DataLoader, device=None):
        self.data_loader = data_loader
        self.iter = iter(self.data_loader)
        self.device = device

    def __next__(self):
        try:
            data = next(self.iter)
            if self.device is not None:
                data = send_to_device(data, self.device)
        except StopIteration:
            self.iter = iter(self.data_loader)
            data = next(self.iter)
            if self.device is not None:
                data = send_to_device(data, self.device)
        return data

    def __len__(self):
        return len(self.data_loader)
```

Audio Augmentation을 위해 RawBoost, Wavaugment 등 다양한 기법을 시도해 보았으나 유의미한 성능 향상을 보여주지 못하여 사용하지 않았습니다.

AASIST

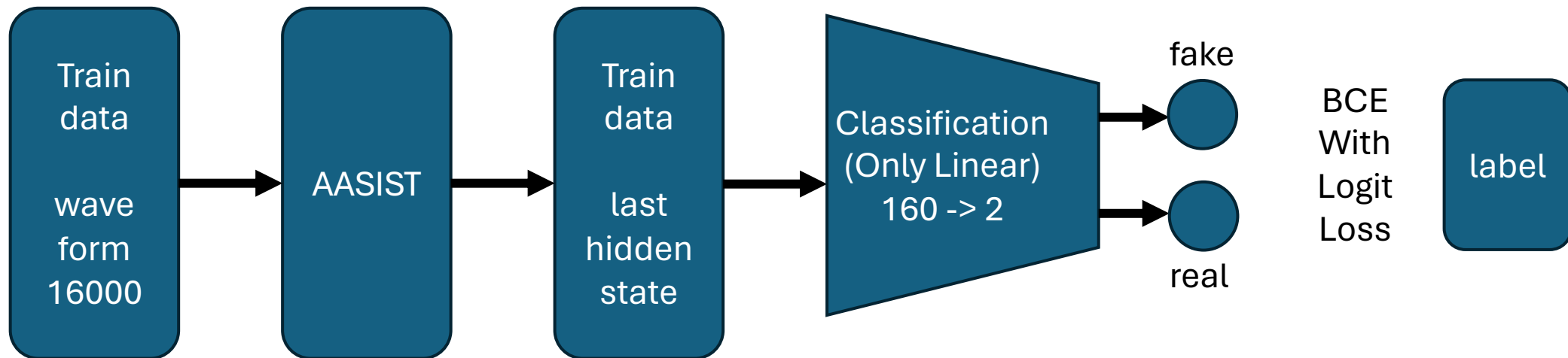


AASIST: AUDIO ANTI-SPOOFING USING INTEGRATED SPECTRO-TEMPORAL GRAPH ATTENTION NETWORKS

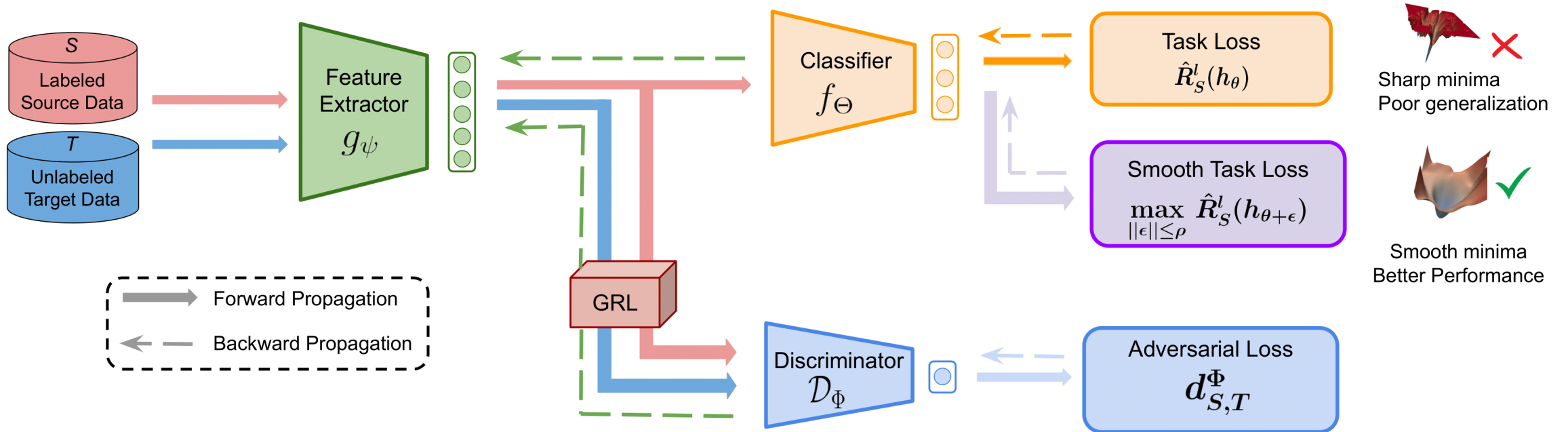
Jee-weon Jung , Hee-Soo Heo , Hemlata Tak , Hye-jin Shim , Joon Son Chung , Bong-Jin Lee , Ha-Jin Yu , Nicholas Evans

<https://arxiv.org/pdf/2110.01200>

General Training AASIST



Smooth Domain Adversarial Training



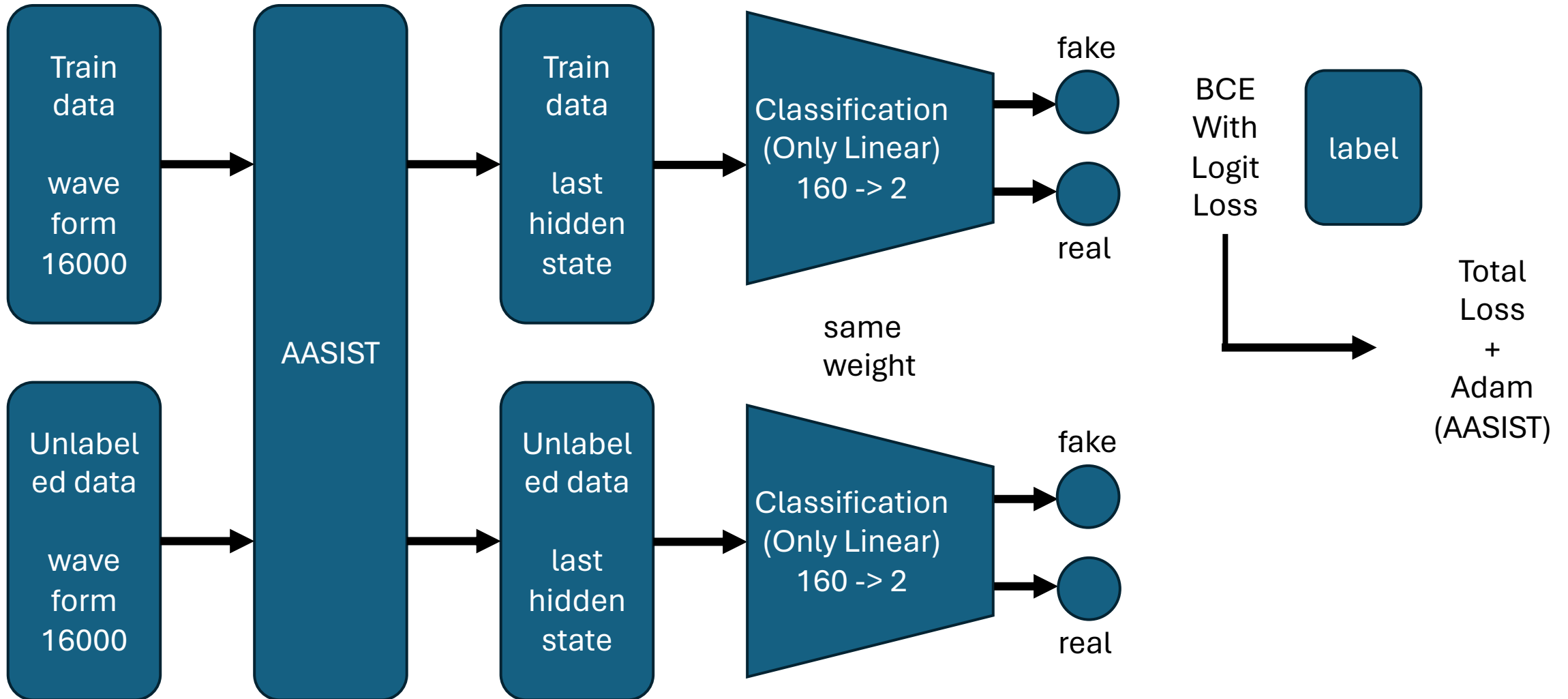
ICML 2022

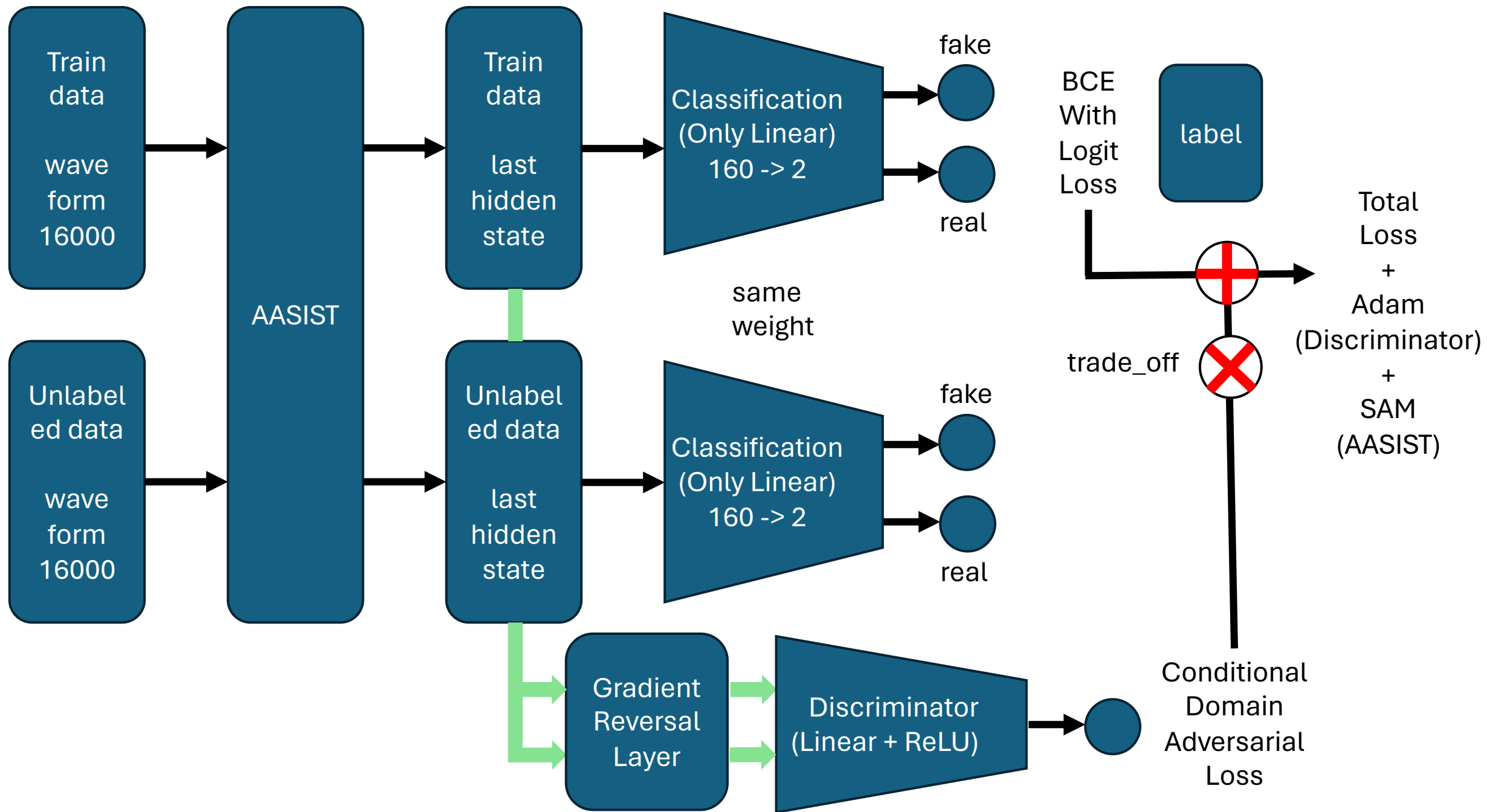
title={A Closer Look at Smoothness in Domain Adversarial Training}

author={Rangwani, Harsh and Aithal, Sumukh K and Mishra, Mayank and Jain, Arihant and Babu, R. Venkatesh}

<https://arxiv.org/abs/2206.08213>

Training AASIST(1)





Discriminator model structure

Index	layer	Input	Output
0	Linear	160(hidden state size) * 2(output size)	1024(hyperparameter)
1	BatchNorm1D	1024	1024
2	ReLU	1024	1024
3	Linear	1024	1024
4	BatchNorm1D	1024	1024
5	ReLU	1024	1024
6	Linear	1024	1
7	sigmoid	1	1

Training setting

batch size : 8

epoch : 100

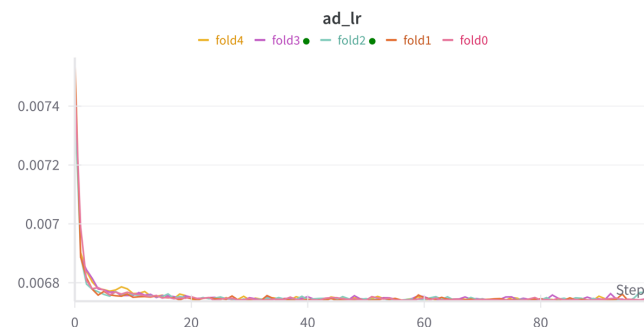
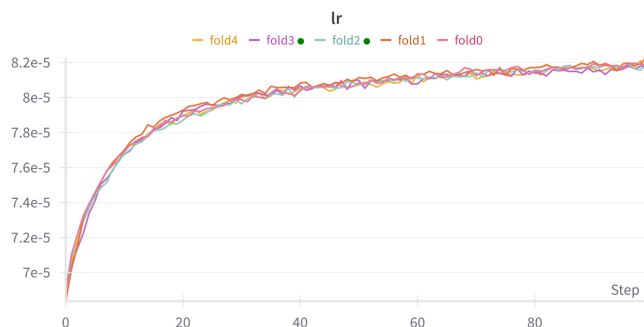
steps per epoch : 1000

weight decay : 0.001

label smoothing : 0.1

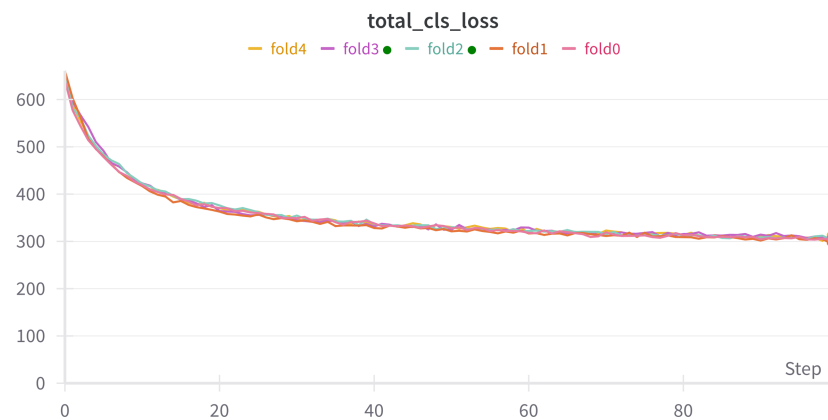
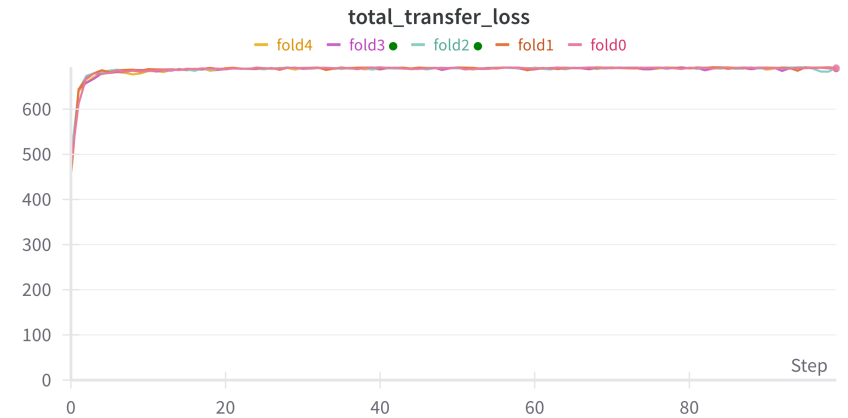
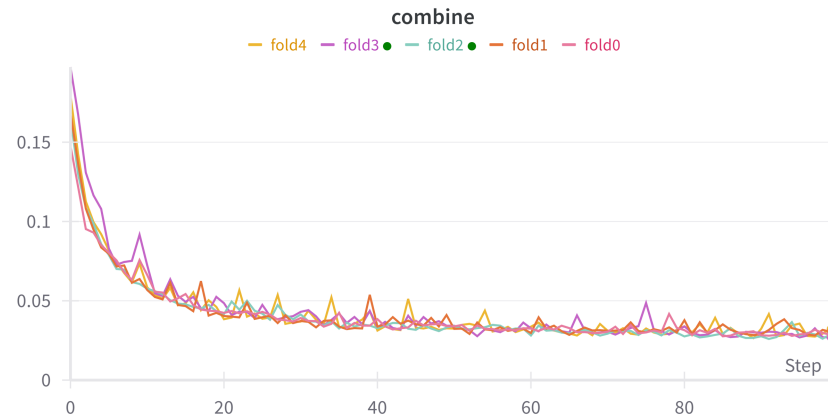
learning rate scheduler : $\text{LambdaLR}(lr * (1. + 0.001 * x)^{-\frac{3}{4}})$

$x = \text{loss}$

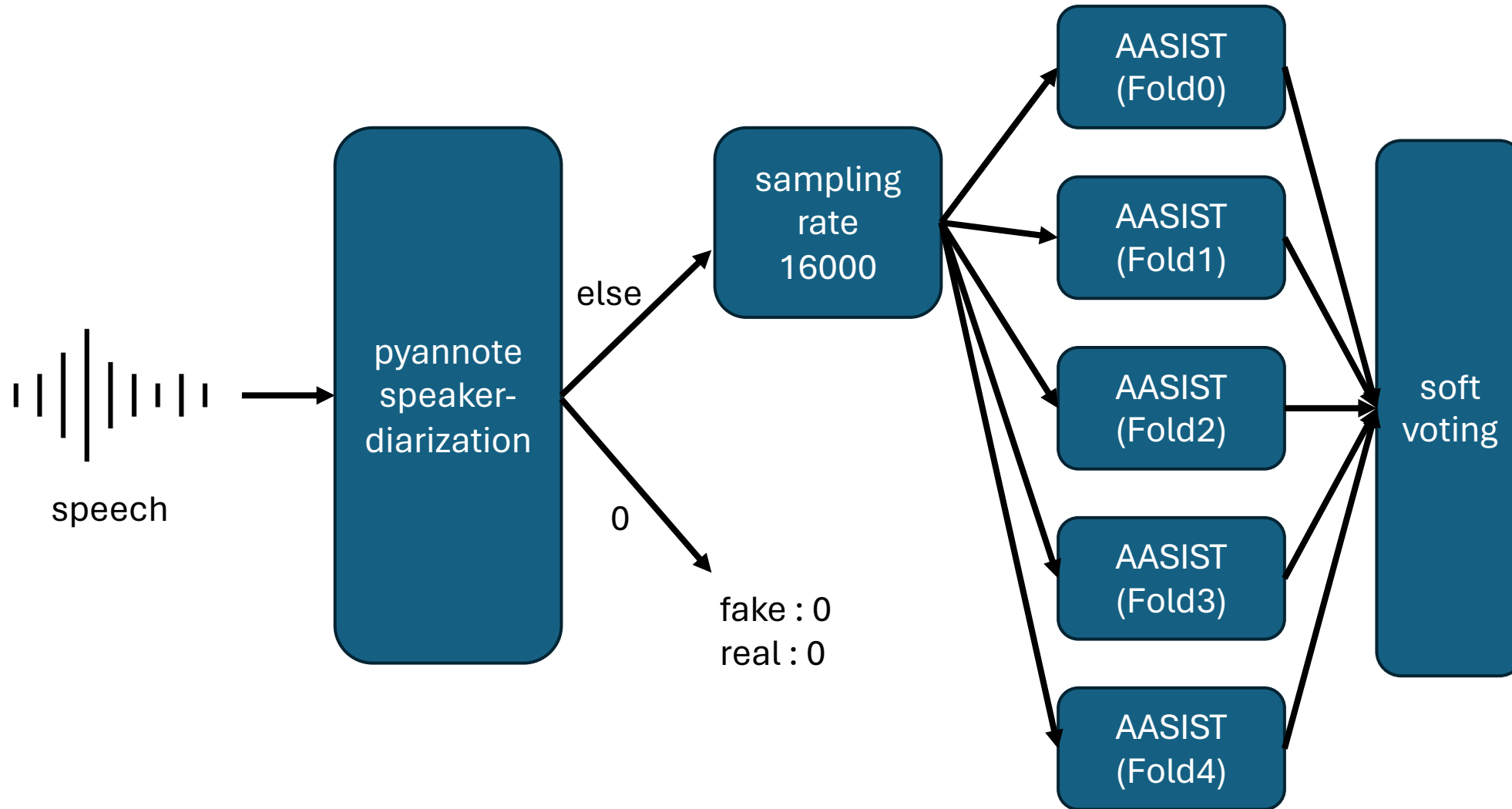


Evaluation of the Results

Stratified 5-fold

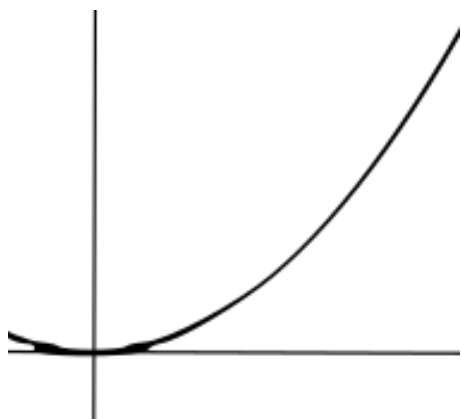


Last model structure(Inference)



understand of eval metric

$$\text{Brier Score} = \frac{\sum_{i=1}^N (p_i - y_i)^2}{N}$$



y의 값은 0 또는 1임.
모델의 accuracy가 낮을 때, model의 confidence를 낮춰주면 일반적으로 더 좋은 score을 얻게 됨.

ECE(Expected Calibration Error)

1. 구간 나누기: 예측 확률을 M 개의 구간으로 나눕니다.
2. 구간 내 평균 예측 확률: 각 구간 m 에 대해 평균 예측 확률 $\text{conf}(m)$ 을 계산합니다.
$$\text{conf}(m) = \frac{1}{|B_m|} \sum_{i \in B_m} p_i$$
3. 구간 내 실제 결과 비율: 각 구간 m 에 대해 실제 결과의 평균 $\text{acc}(m)$ 을 계산합니다.
$$\text{acc}(m) = \frac{1}{|B_m|} \sum_{i \in B_m} y_i$$
4. ECE 계산:
$$\text{ECE} = \sum_{m=1}^M \frac{|B_m|}{N} |\text{acc}(m) - \text{conf}(m)|$$

여기서:
 - B_m = 구간 m 에 속하는 샘플들의 집합
 - $|B_m|$ = 구간 m 의 샘플 수
 - N = 전체 샘플 수

accuracy와 평균 예측 확률이 같다면 해당 score은 0이 됨.
양성 샘플 비율과 비슷한 확률을 가진다면 일반적으로 좋은 score을 얻게 됨.

(주로 calibration이 얼마나 잘 되었는지 평가하는 지표)

Output of unlabeled data

Confidence가 너무 높다

fold0

[0.6175, 0.9288]
[0.9395, 0.7519]
[0.9346, 0.0307]
[0.8987, 0.8047]
[0.8901, 0.8763]
[0.9212, 0.9168]
[0.9356, 0.0804]
[0.1516, 0.8994]

fold1

[0.9279, 0.9121]
[0.9457, 0.4829]
[0.9434, 0.0102]
[0.8936, 0.5888]
[0.9465, 0.5995]
[0.9476, 0.7853]
[0.9371, 0.0385]
[0.4750, 0.9066]

fold2

[0.7853, 0.9263]
[0.9324, 0.7324]
[0.9415, 0.1179]
[0.7492, 0.8623]
[0.8862, 0.8847]
[0.8352, 0.9149]
[0.9374, 0.2076]
[0.0545, 0.9035]

fold3

[0.8538, 0.9358]
[0.9461, 0.6750]
[0.9223, 0.0301]
[0.9289, 0.7726]
[0.9262, 0.8201]
[0.9322, 0.9171]
[0.9125, 0.0837]
[0.0261, 0.9251]

fold4

[0.8959, 0.9238]
[0.9433, 0.5732]
[0.9560, 0.0085]
[0.8786, 0.8196]
[0.9260, 0.7095]
[0.9418, 0.6047]
[0.9414, 0.0477]
[0.1902, 0.9089]

Postprocessing

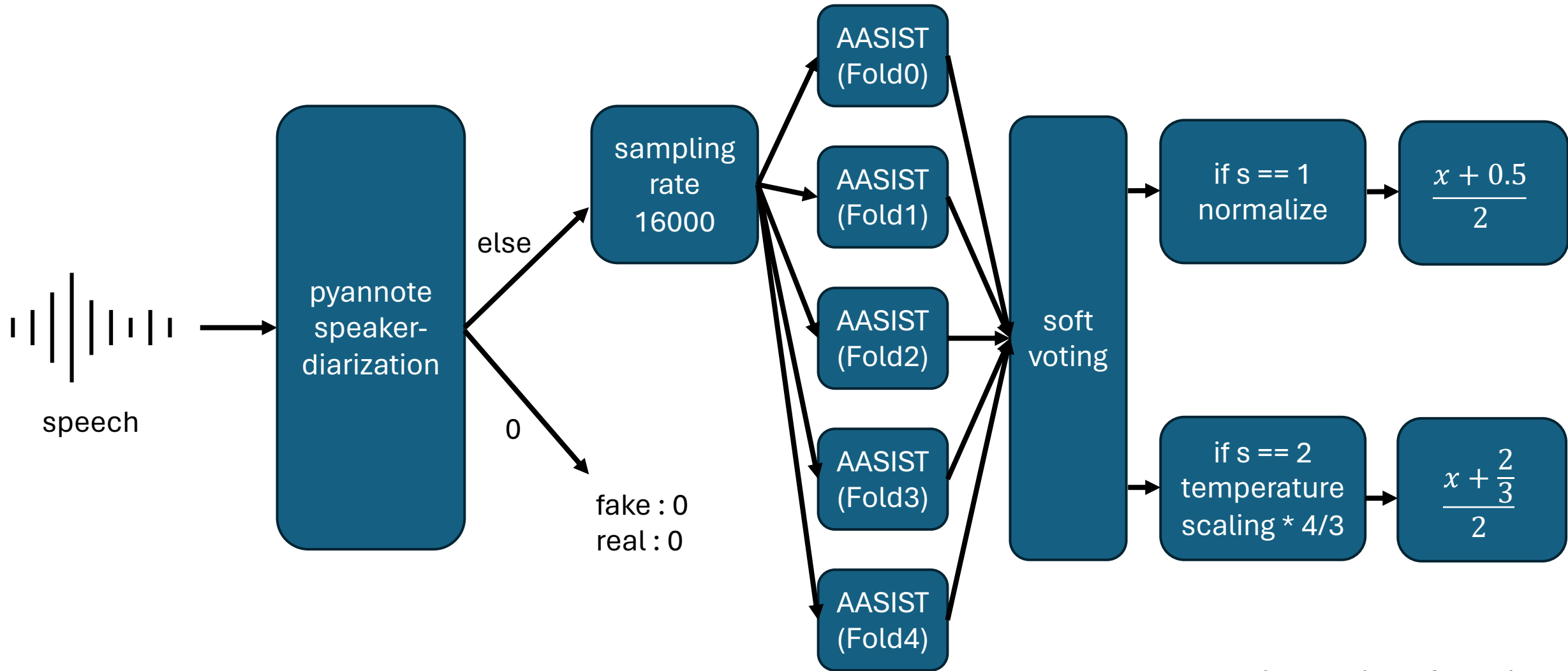
normalize

```
def normalize(x):  
    sum_x = np.sum(x, axis=1, keepdims=True)  
    return x / sum_x
```

temperature_scaling

```
def temperature_scaling(logits, temperature=1.1):  
    scaled_logits = logits / temperature  
    exp_logits = np.exp(scaled_logits)  
    return exp_logits / np.sum(exp_logits, axis=1, keepdims=True)
```


Last model structure(Inference) - modify



s = the number of speaker

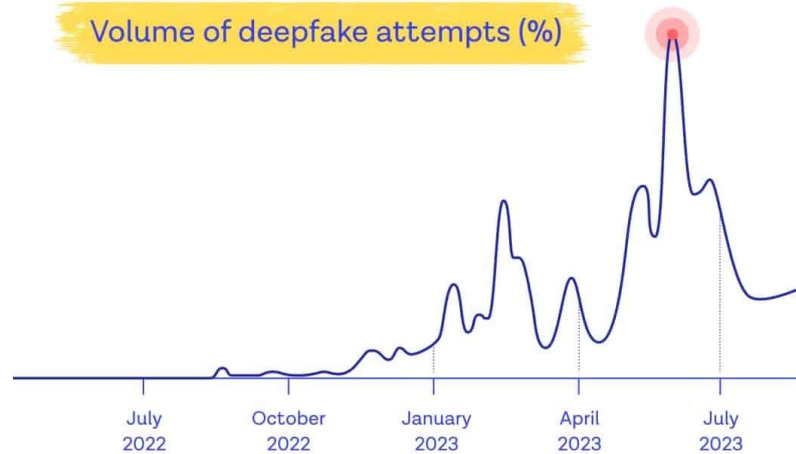
Deployment

sr = 16000

unsupervised learning(domain adaptation)

-> 현업에서 빠르게 적용 가능, 일반화

Volume of deepfake attempts (%)



onfido Identity Fraud Report 2024

<https://onfido.com/landing/identity-fraud-report/>

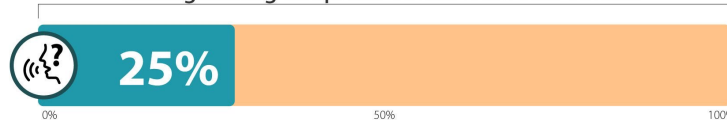
저희의 모델은 시간이 지남에 따라 더욱 많아질 deepfake fraud를 예방하게 해주고,

audio에 deepfake가 있는지 없는지 판단해 사람들에게 도움이 될 수 있도록 하고,



Difficulty in Identifying Deepfake Audio

Percentage of people who have difficulty distinguishing deepfake voices from real voices.



Source: LocalCircles (2023)

<https://keepnetlabs.com/blog/deepfake-statistics-and-trends-about-cyber-threats-2024>



Public Unawareness and Misconception

People surveyed around the world.

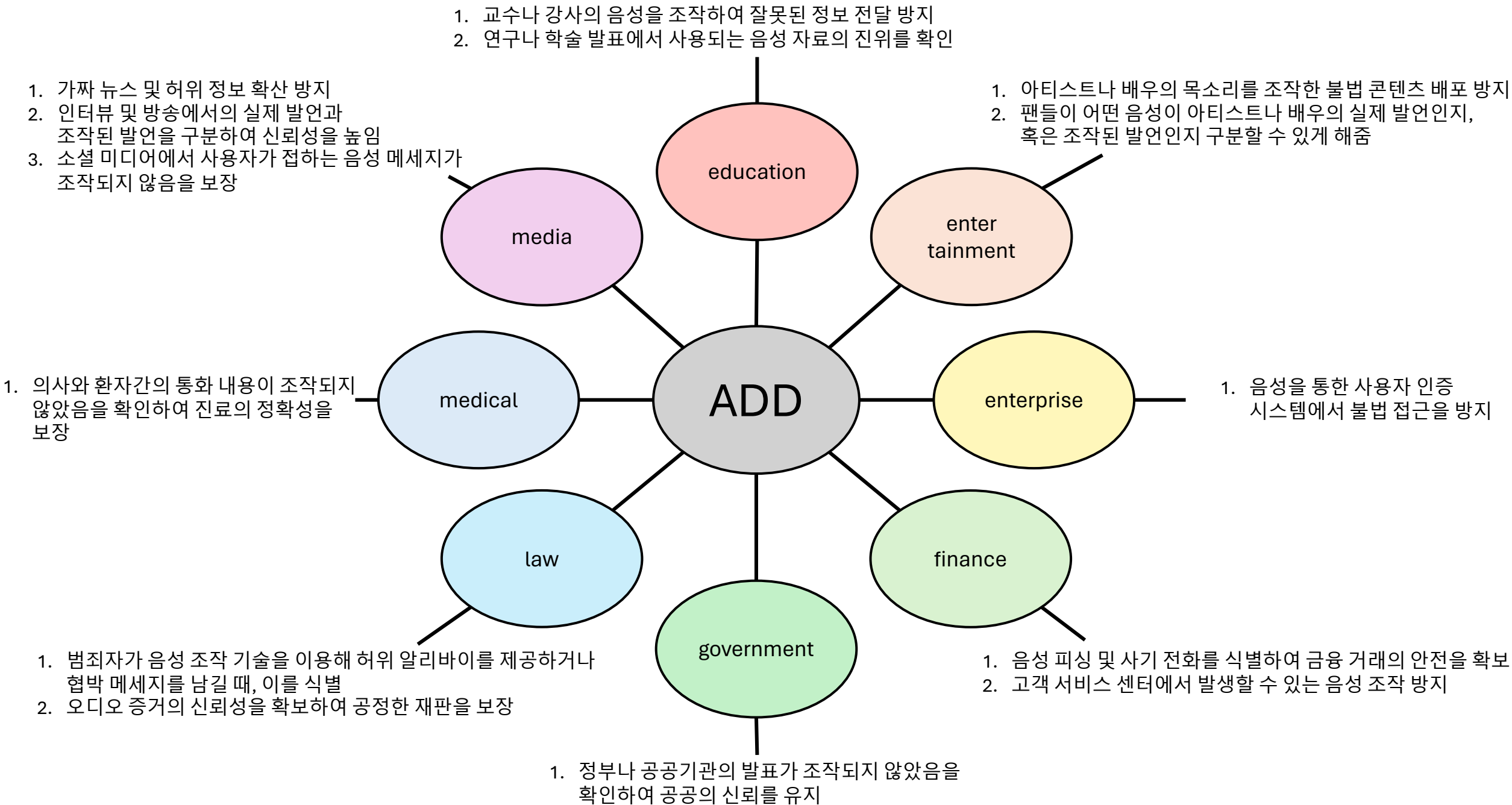


Source: Iproov

keepnet

<https://keepnetlabs.com/blog/deepfake-statistics-and-trends-about-cyber-threats-2024>

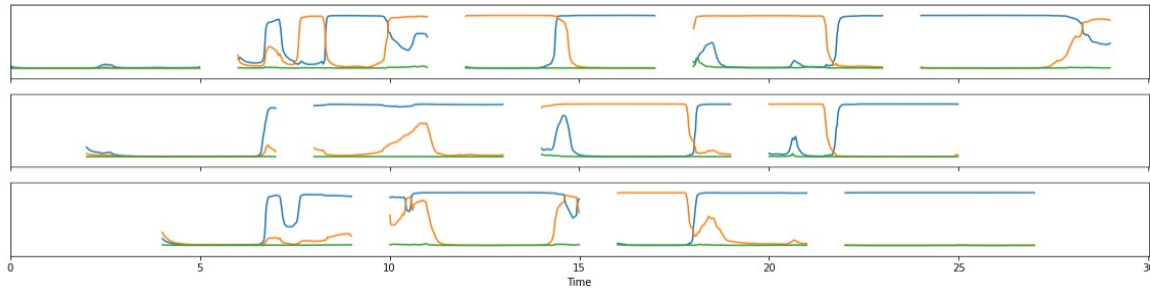
deepfake audio가 의심된다면 deepfake가 무엇이고 의심된다고 말해줄 수 있을 것입니다.



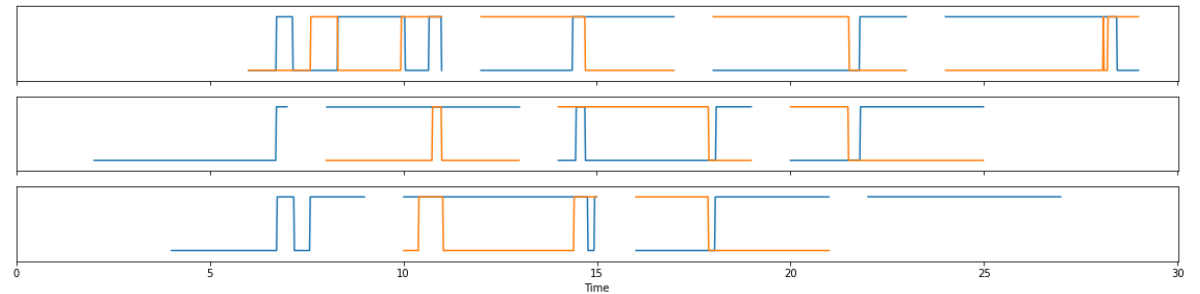
Thank you

pyannote/speaker-diarization-3.1 (1)

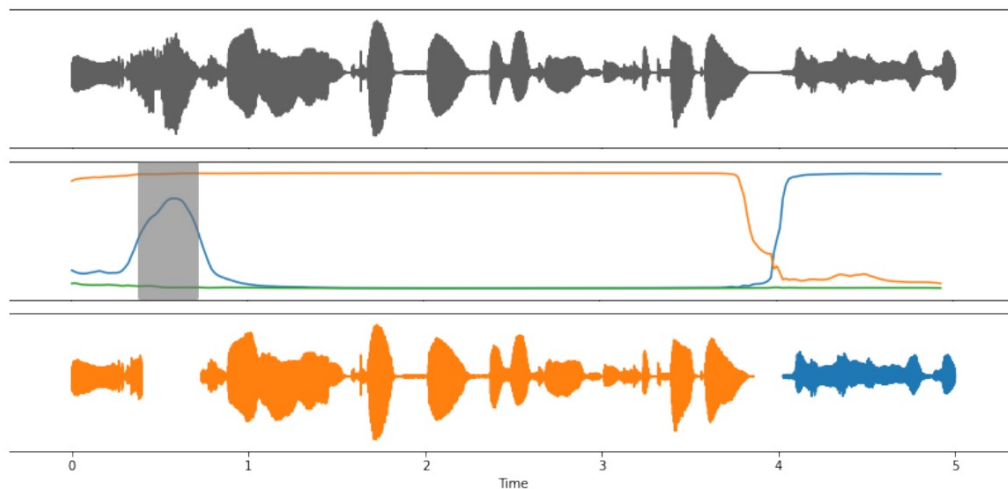
1. local neural speaker segmentation



2. binary local speaker segmentation

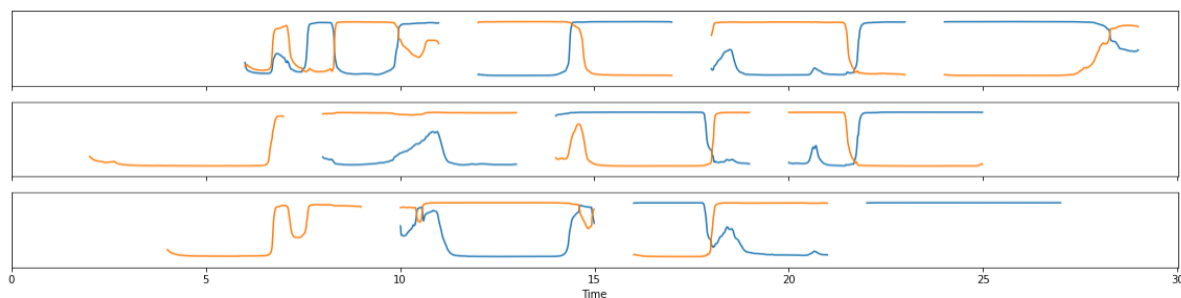


3. Local speaker embedding



pyannote/speaker-diarization-3.1 (2)

4. Global agglomerative clustering



5. Final aggregation

