



캡스톤

20222016



## 문제 정의 - Deepfake Detection

---

### 1. 딥페이크란

- 딥러닝 + 페이크의 합성어로 인공지능 기술을 활용하여 사람의 얼굴, 목소리, 영상 등을 합성하는 기술
- 주로 GAN(생성형 적대 신경망)을 사용해 사실적인 이미지를 생성

### 2. 과정

- 얼굴 탐지 : 입력 영상에서 얼굴을 식별
- 얼굴 교환 : 특정 얼굴을 다른 인물로 합성
- 후처리 : 자연스러운 연결을 위한 후보정



## 딥페이크 이미지 예시

---

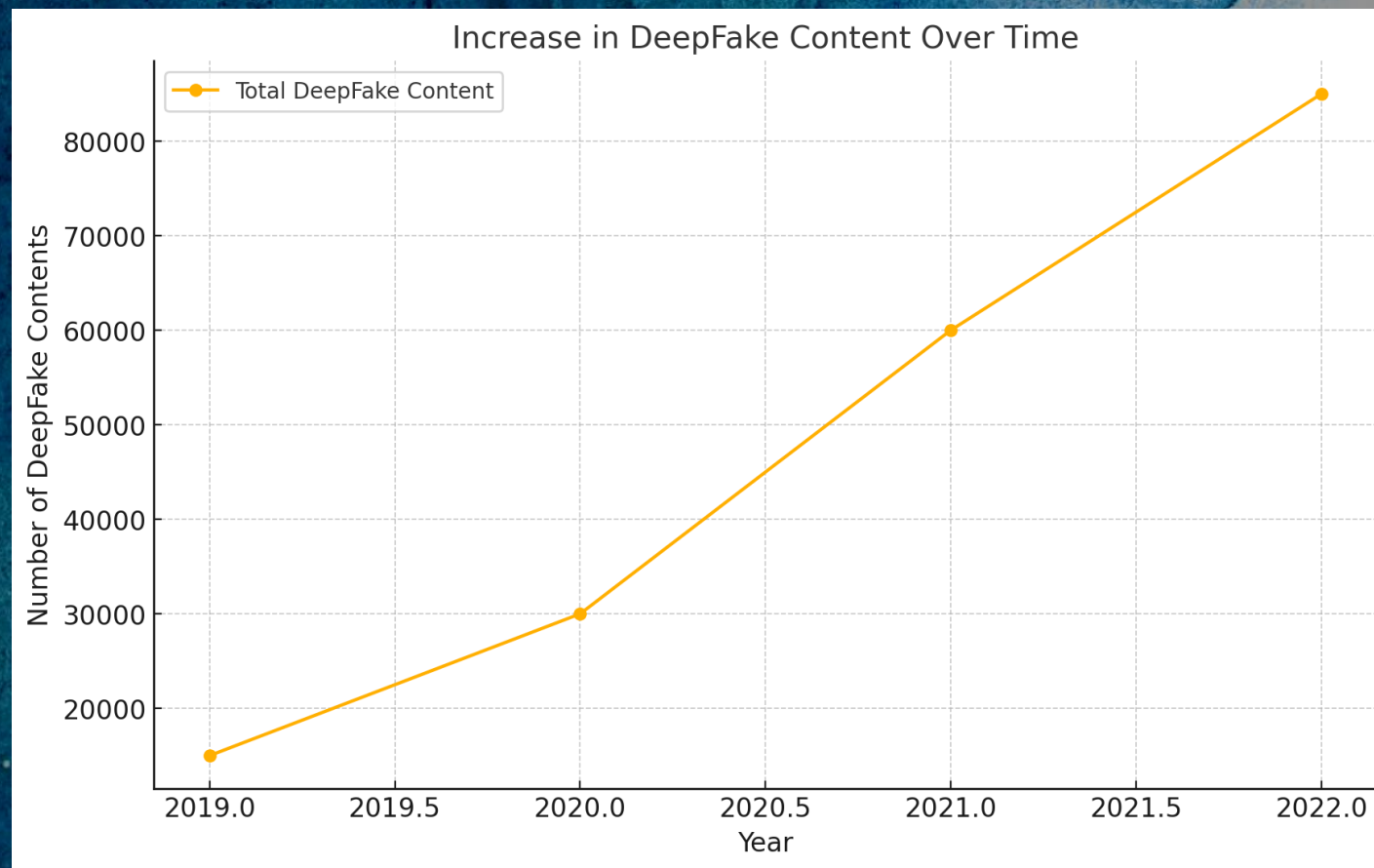




## 딥페이크 탐지 연구의 필요성

---

1. 사회적 신뢰의 위협
2. 사이버 범죄의 증가
3. 산업 및 상업적 피해
4. 정치적 안정성과 국가 안보
5. 기술의 양면성
6. 법적 및 윤리적 요구
7. 탐지 기술의 발전



위 그래프는 딥페이크 콘텐츠의 증가 추세를 나타냄



# 딥페이크 탐지 플랫폼들

---

## 1. Microsoft - Video Authenticator

- 비디오 및 이미지의 진위 여부 확인
- 픽셀 수준의 세부 정보를 분석하여 영상의 변조 가능성을 점수로 표시
- 프레임 단위 분석하여 딥페이크 여부를 판별

## 2. Deepware Scanner

- 딥페이크 탐지 전용 모바일 앱
- 음성 및 비디오 데이터를 분석하여 진위 여부를 확인
- 무료 사용, 실시간 탐지 가능

## 3. Sensity

- 딥페이크 탐지를 위한 API 및 플랫폼 제공
- 대규모 데이터와 AI를 사용해 고급 딥페이크 탐지 수행

## 4. FaceForensics++ ( 딥페이크 탐지 관련 데이터 셋)



## 딥페이크 이미지 탐지에 사용되는 모델들

→ 2019 ~ 2022 연구 취합 논문 출처

**TABLE 5. Distribution of used models.**

Category	Model	#Studies	PCT (%)
Deep Learning	CNN	71	78%
	RNN	12	13%
	RCNN	2	2%
Machine Learning	SVM	11	12%
	k-MN	4	4%
	LR	3	3%
	MLP	3	3%
	BOOST	2	2%
	RF	1	1%
	DT	1	1%
	DA	1	1%
	NB	1	1%
	MIL	1	1%
Statistical	EM	1	1%
	TV, KL, JS	1	1%



## 딥페이크 이미지 탐지에 사용되는 데이터셋들

---

Database Name	#Deepfakes	#Actors
FaceForensics (FF) [11]	1000	977
FaceForensics++ (FF++) [42]	1000	977
DeepfakeDetection (DFD) [135]	3000	28
DeepFake Forensics (Celeb-A) [136]	202,599 (images)	10,177
DeepFake Forensics (Celeb-DF) [137]	795+ 590	13+59
Deepfake Detection Chal. (DFDC) [138]	5214	66
UADFV [27]	49	-
Deepfake-TIMIT (DF-TIMIT) [91]	620	64
DeeperForensics-1.0 (DF-1.0) [139]	60,000	100
WildDeepfake (WDF) [140]	707	100
MANFA [84]	8950 (images)	-
SwapMe and FaceSwap (SMFW) [30]	1005 (images)	-
Deep Fakes (DFS) [22]	142	-
Fake Faces in the Wild (FFW) [141]	150	-
FakeET (FE) [142]	811	40
FaceShifter (FS) [143]	5000 (images)	-
Deepfake (DF) [39]	175	50
Swapped Face Detection (SFD) [81]	420,053 (images)	86

→ 2019 ~ 2022 연구 취합 논문 출처



## 딥페이크 이미지 탐지에 사용되는 일반적인 특징들

---

1. 조명 불일치 - 얼굴의 일부가 부자연스럽게 밝거나 어둡다
2. 경계선 블러 효과 - 얼굴 윤곽이 흐릿하거나 배경과 부드럽게 섞임
3. 입 주변 아티팩트 - 말할 때의 입술 움직임에서의 깨짐 및 이상한 픽셀화
4. 머리카락과 배경의 왜곡 - 세밀한 부분이 부자연스럽거나 흐릿하게 보임



## 딥페이크 이미지 탐지하는 과정 및 방법

---

1. 인공지능 모델을 통해 딥페이크 이미지의 특징 추출

2. 특징 추출 방법 (= 학습 방법)

- (1) 이미지를 RGB (0,225)값으로 변환
- (2) 정규화 및 전처리
- (3) 커널 (3X3 , 5X5)을 통해 이미지 특성 학습
- (4) 풀링층, 활성화 함수를 거쳐 예측 반환

→ 각 라벨의 특징을 추출해 딥페이크인지 아닌지 이진 분류 결과값(=확률값)을 통해 탐지



## 딥페이크 오디오

---

- 딥페이크 오디오란 인공지능 기술을 활용하여 특정 인물의 목소리를 정교하게 복제
- 실제와 거의 구분할 수 없는 자연스러운 음성 생성
- 감정, 억양, 속도, 발음까지 복제 가능
- 단 몇 초의 샘플 음성만으로도 학습 가능
- voice scam 이나 정치적 조작 등 불법적인 곳에 사용될 수 있음





## 딥페이크 오디오 와 실제 오디오의 차이점

---

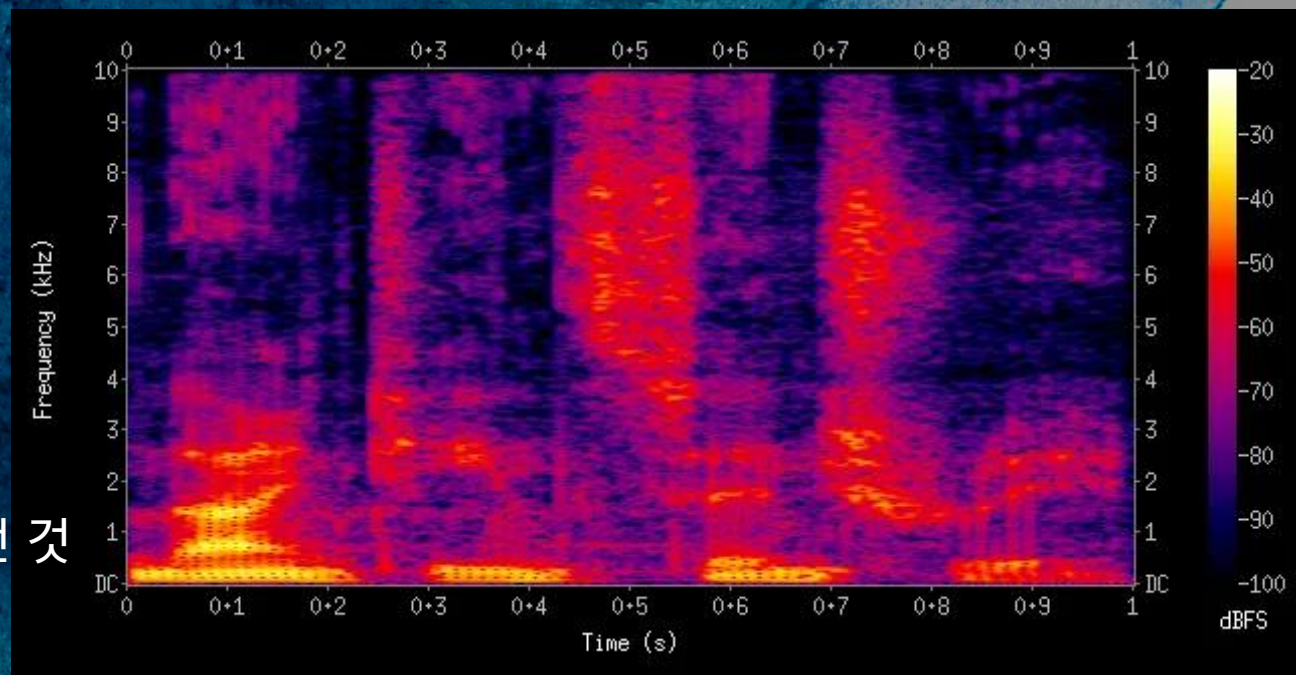
1. 사람의 음성은 자연스러운 주파수 변동, 고유한 스펙트럼 유지
2. 딥러닝으로 생성된 음성은 특정 주파수 대역에서 부자연스러운 왜곡이 나타남
3. 고주파수 대역(4kHz 이상)에서 디테일 부족, 배경 잡음이 비현실적이나 일정한 패턴을 가짐
4. 스펙트로그램으로 실제 음성과 딥페이크 음성의 주파수 패턴을 비교
5. MFCCs 음성의 주파수 특성을 벡터화하여 머신러닝 모델로 분석



## 딥페이크 오디오 탐지 방법 - 스펙트로그램

---

1. 오디오 파일을 통해 스펙트로그램으로 전환 →
2. 스펙트로그램은 소리나 파동을 그래프로 나타낸 것
3. 스트로그램으로 전환 후 이미지 분류와 같은 방법으로 분류





## 딥페이크 오디오 탐지 방법 - MFCC 기반 딥페이크 탐지

---

1. MFCC는 시계열 데이터, LSTM, GRU, Transformer같은 RNN 계열 모델이 적합
2. 사람의 청각은 선형적인 주파수 인식이 아닌 Mel Scale이라는 비선형 스케일을 따름
3. MFCC는 이러한 인간 청각의 특성을 반영해 음성 신호를 벡터화하여 머신러닝이 학습할 수 있도록 함
4. 특징 벡터화한 데이터를 수치화해서 목적에 맞게 인공지능 모델에 학습시키기 위한 필수 과정
5. 오디오를 추출한 다음 10~40ms 단위로 끊어 각 구간 별 주파수를 매기는 방식
6. 딥페이크 오디오는 목소리를 씌우는 방식으로 주로 배경 소리에 변화가 있음
7. 이를 통해 딥페이크 오디오를 탐지 ( 그 전의 주파수와 현재의 주파수 비교를 통한 딥러닝 모델 학습)



## 딥페이크 탐지 계획 구축

---

1. 데이터 셋 준비 (이미지 + 오디오)
2. 모델 준비
3. 모델 학습
4. 학습 정보 저장
5. 딥페이크 관련 실제 데이터 삽입 및 테스트
6. 디버깅



## 딥페이크 탐지 원본 데이터셋

---

### REAL-TIME DETECTION OF AI-GENERATED SPEECH FOR DEEPPAKE VOICE CONVERSION

(Jordan J. Bird, Ahmad Lotfi

Nottingham Trent University Nottingham, UK {jordan.bird, [ahmad.lotfi@ntu.ac.uk](mailto:ahmad.lotfi@ntu.ac.uk)}


- MIT 라이선스 하에 제공 (상업적 이용 가능)
- 위 논문에서 구축한 데이터셋
- Label : REAL ( 실제 사람의 음성), FAKE( RVC 기법을 사용해 변환된 딥페이크 음성)
- <https://arxiv.org/pdf/2308.12734>

### Open Forensics

- 고품질의 얼굴 합성 탐지 및 분할을 위한 대규모 데이터셋
- 라이선스 : Creative Commons Attribution 4.0 (CC BY 4.0)
- 저작권 표시 조건으로 상업적 이용 가능함



# 1. 딥페이크 이미지 데이터셋

 MANJIL KARKI · UPDATED 3 YEARS AGO

73

<> Code

Download

## deepfake and real images

Detect if any images is real image of deepfake image

[Data Card](#) [Code \(150\)](#) [Discussion \(0\)](#) [Suggestions \(0\)](#)



## CelebA-HQ resized (256x256)

The CelebA-HQ celebrity faces dataset



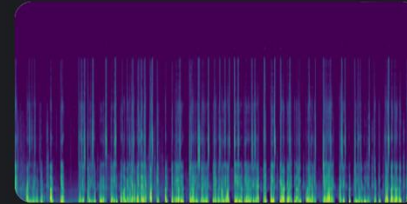
[Data Card](#) [Code \(30\)](#) [Discussion \(3\)](#) [Suggestions \(0\)](#)



# 1. 딥페이크 오디오 데이터셋

## DEEP-VOICE: DeepFake Voice Recognition

Using machine learning to detect when speech is AI-Generated



Data Card

Code (54)

Discussion (3)

Suggestions (0)

### About Dataset

Usability ⓘ

10.00

## The Fake-or-Real (FoR) Dataset (deepfake audio)

dataset to detect synthetic speech



Data Card

Code (13)

Discussion (0)

Suggestions (0)

### About Dataset

Usability ⓘ

7.50



## 2. 모델 준비 – EfficientNet

---

1. EfficientNet은 구글에서 발표한 compound scaling으로 유명한 모델
2. compound scaling이란 기존의 네트워크 너비 깊이, 해상도 모두 키우는 방식의 모델
3. 모델의 무게에 따라 B0부터 B7까지 모델이 나뉨
4. 기존의 성능 향상 방법으로는 해상도, 깊이, 너비 중 한두가지만 키우는 방식이 대다수
5. compound scaling을 통해 성능 향상을 효과적으로 이루어 냄
6. Apache2.0으로 출처만 밝힌다면 상업적 사용도 허가함



## 2. 모델 준비 - LSTM

---

1. 시계열 데이터를 위한 모델
2. 탐지 대상 오디오를 10ms ~ 40ms 으로 쪼개어 MFCC 특징 추출
3. 이것을 시계열 데이터라고 하고 모델은 시간의 흐름에 따른 MFCC 특징 변화를 통해 딥페이크 오디오 탐지
4. 자연어처리(NLP), 음성인식 등에 많이 사용함
5. Forget Gate, Input Gate, Output Gate 를 통해 중요 정보 필터링



### 3. 모델 학습 - 이미지 전처리

---

```
def detect_fake_image(model, image_path):  
    """ 단일 이미지 기반 딥페이크 탐지 """  
    transform = transforms.Compose([  
        transforms.Resize((300, 300)),  
        transforms.ToTensor(),  
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),  
    ])  
    img = Image.open(image_path).convert("RGB")  
    img = transform(img).unsqueeze(0)
```

→ 이미지 300 \* 300 픽셀화 → efficientnet 에 맞는 스케일링

→ 각 픽셀 별 RGB값 변환 후 정규화 → 색상 정보가 중요한게 아닌 마스크의 텍스처(변화)가 중요



### 3. 모델 학습 → EfficientNet -B3

---

- CrossEntropyLoss 손실함수 사용
  - triplet을 사용하려 했지만 로스값이 1밑으로 떨어지지 않아 선택
  - 주로 이진분류, 다중분류에서 많이 사용하는 손실함수
  - 경사하강법으로 미분가능하고 연속적인 함수로 분류 문제에 적합

→ optimizer - AdamW

→ 모델에 맞게 300 x 300 픽셀 사용

→ 이미지 정규화를 통해 색상정보는 영향을 최소화



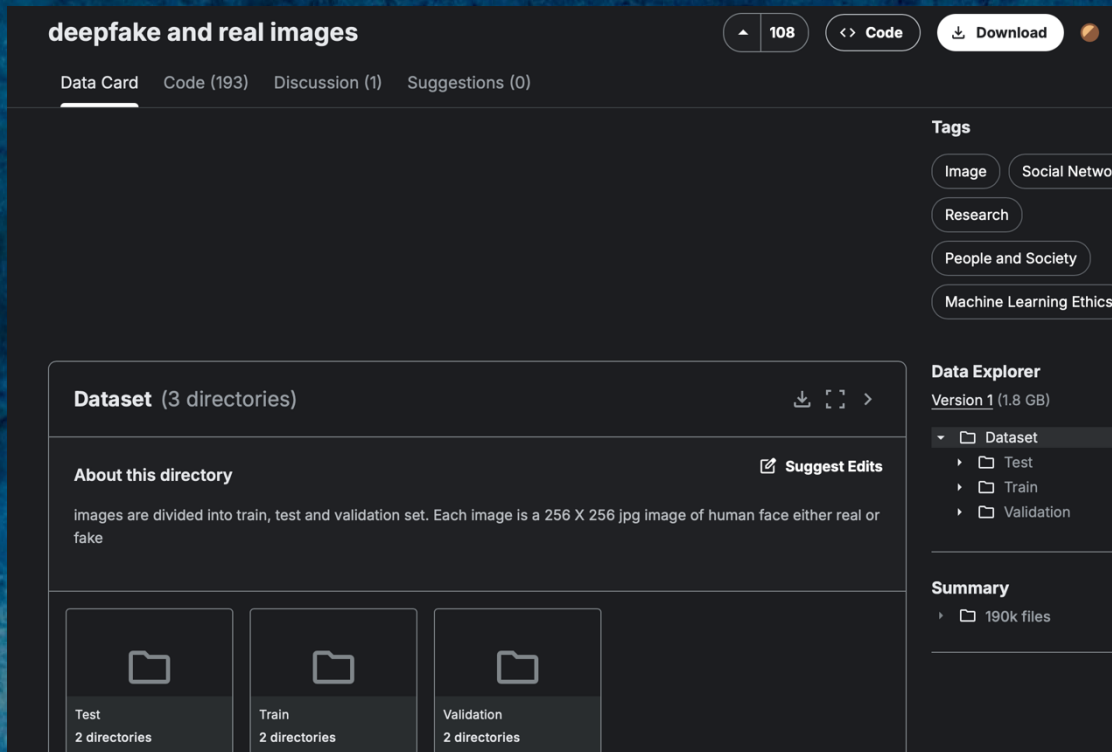
# 첫번째 데이터셋

딥페이크 이미지와 실제 이미지로 이루어진  
총 15만장의 데이터셋

출처: openforensics



<https://zenodo.org/record/5528418#.YpdIS2hBzD>



The screenshot shows the Zenodo dataset page for 'deepfake and real images'. The page has a dark theme. At the top, there's a header with the dataset name 'deepfake and real images', a version indicator '108', and buttons for 'Code' and 'Download'. Below the header, there are tabs for 'Data Card', 'Code (193)', 'Discussion (1)', and 'Suggestions (0)'. The 'Data Card' tab is active. The main content area is divided into two columns. The left column contains a 'Dataset (3 directories)' section with a download icon and a 'Data Explorer' section showing a tree view of the dataset structure: 'Dataset' (expanded) containing 'Test', 'Train', and 'Validation' directories. The right column contains a 'Summary' section showing '190k files'. Below the 'Dataset (3 directories)' section, there are three cards representing the dataset directories: 'Test' (2 directories), 'Train' (2 directories), and 'Validation' (2 directories). Each card has a folder icon and text indicating the number of directories.



# 두번째 데이터셋

The screenshot shows the Hugging Face interface for the 'deepfake and real images' dataset. At the top, the title 'deepfake and real images' is displayed, followed by a 'Data Card' tab and buttons for 'Code' and 'Download'. Below the title, there are tabs for 'Data Card', 'Code (193)', 'Discussion (1)', and 'Suggestions (0)'. The main content area is divided into two sections: 'Dataset (3 directories)' and 'Data Explorer'. The 'Dataset' section includes a description: 'images are divided into train, test and validation set. Each image is a 256 X 256 jpg image of human face either real or fake'. Below this, there are three folder icons representing 'Test', 'Train', and 'Validation' directories, each containing 2 directories. The 'Data Explorer' section shows the dataset structure: 'Dataset' (1.8 GB) containing 'Test', 'Train', and 'Validation' folders. A 'Summary' section indicates there are 190k files.

deepfake and real images

108 Code Download

Data Card Code (193) Discussion (1) Suggestions (0)

Tags

- Image
- Social Network
- Research
- People and Society
- Machine Learning Ethics

Dataset (3 directories)

About this directory [Suggest Edits](#)

images are divided into train, test and validation set. Each image is a 256 X 256 jpg image of human face either real or fake

Test 2 directories

Train 2 directories

Validation 2 directories

Data Explorer

Version 1 (1.8 GB)

- Dataset
  - Test
  - Train
  - Validation

Summary

- 190k files

딥페이크 이미지와 실제 이미지 총 15000여장의 데이터셋  
총 400개의 동영상으로부터 프레임 추출

출처: FaceForensics++, Celeb-df



Epoch [1/10], Loss: 275.2458, Train Acc: 0.9754  
Validation Acc: 0.9088, F1 Score: 0.9086  
✓ Best model saved with F1: 0.9086  
Epoch [2/10], Loss: 117.0955, Train Acc: 0.9894  
Validation Acc: 0.9102, F1 Score: 0.9099  
✓ Best model saved with F1: 0.9099  
Epoch [3/10], Loss: 81.9767, Train Acc: 0.9920  
Validation Acc: 0.9197, F1 Score: 0.9193  
✓ Best model saved with F1: 0.9193  
Epoch [4/10], Loss: 62.4752, Train Acc: 0.9944  
Validation Acc: 0.9227, F1 Score: 0.9225  
✓ Best model saved with F1: 0.9225  
Epoch [5/10], Loss: 49.5836, Train Acc: 0.9954  
Validation Acc: 0.9075, F1 Score: 0.9069  
Epoch [6/10], Loss: 41.3206, Train Acc: 0.9963  
Validation Acc: 0.9124, F1 Score: 0.9119  
Epoch [7/10], Loss: 33.4727, Train Acc: 0.9972  
Validation Acc: 0.9185, F1 Score: 0.9183

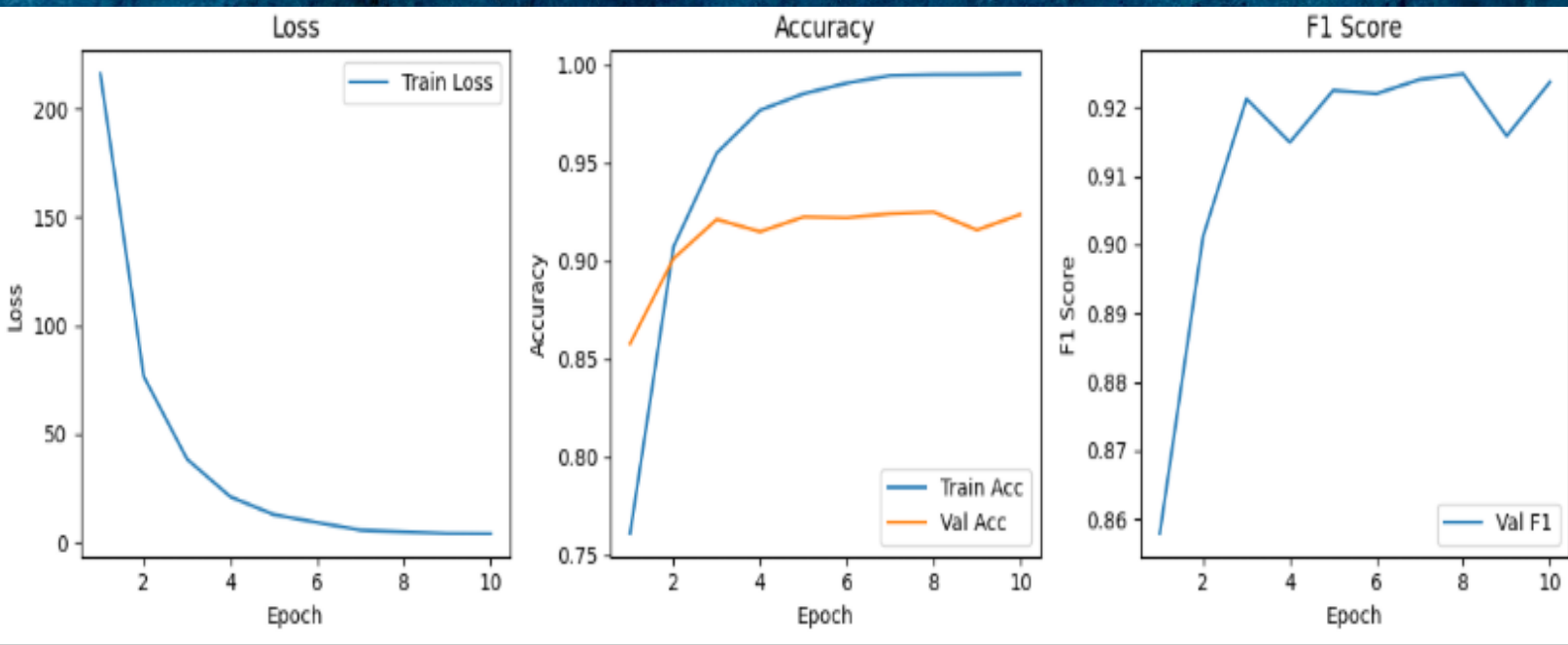
## 첫번째 데이터 훈련결과

- F1 기준으로 베스트 모델 저장
- Loss: 배치사이즈당 Loss값을 모두 더한 것
- 검증셋은 같은 도메인 내 훈련하지 않은 데이터셋



# 두번째 데이터 훈련결과

• 8번째 에포크의 모델 저장





✓ 모델 로드 완료

📊 Test Loss: 0.2886

✓ Test Accuracy: 0.9227

🎯 Test F1 Score: 0.9225

📄 Test LogLoss: 0.2886

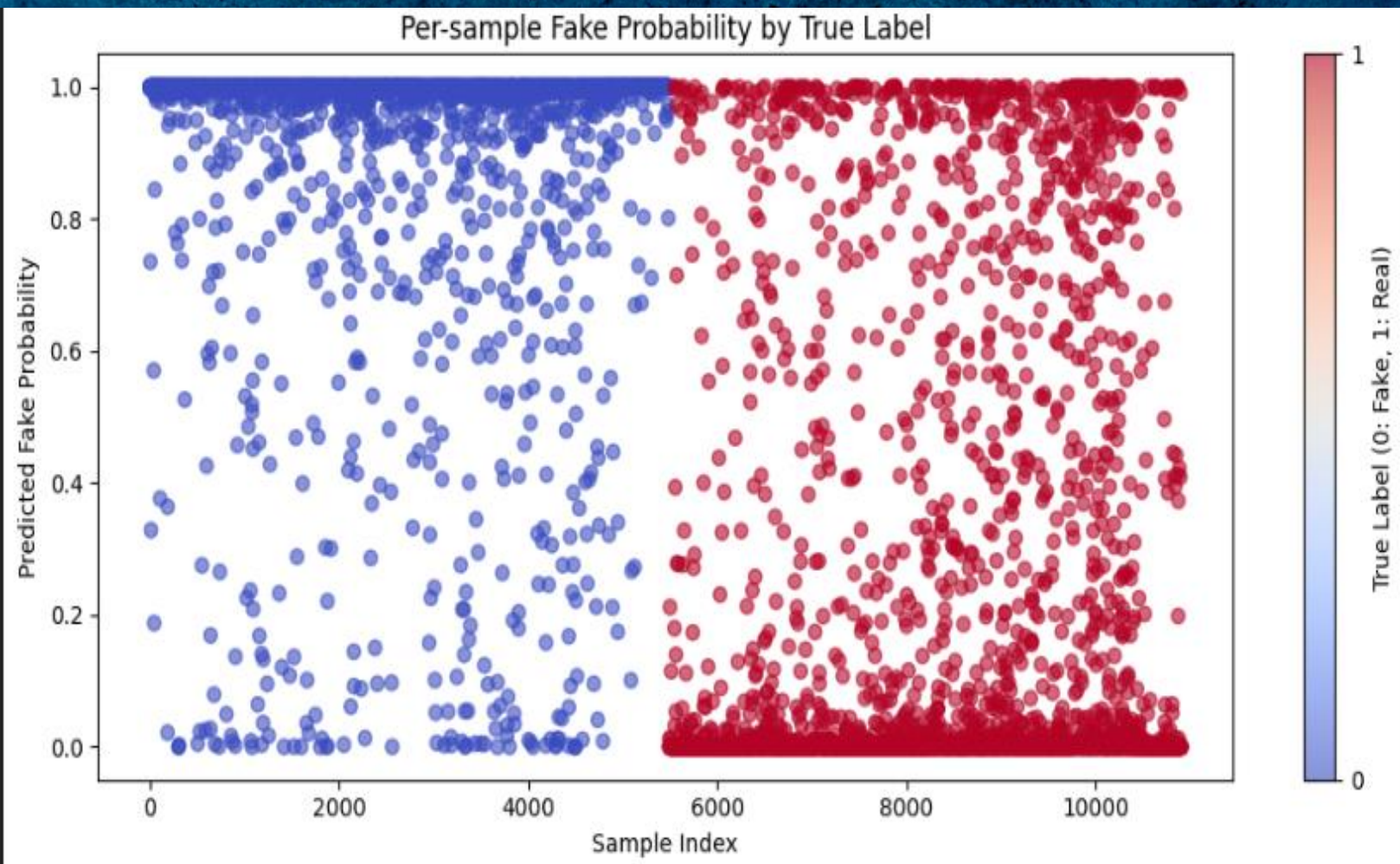
📋 Classification Report:

	precision	recall	f1-score	support
Fake	0.8917	0.9636	0.9262	5492
Real	0.9598	0.8812	0.9188	5413
accuracy			0.9227	10905
macro avg	0.9257	0.9224	0.9225	10905
weighted avg	0.9255	0.9227	0.9225	10905

## 첫번째 데이터 테스트 결과

같은 도메인 내 모델이 학습하지  
않은 데이터셋으로 테스트한 결과





## 첫번째 데이터 테스트 결과

실제 소프트맥스 결과 값의 그래프



✓ 모델 로드 완료

📊 Test Loss: 0.2991

✓ Test Accuracy: 0.9250

🎯 Test F1 Score: 0.9249

📄 Test LogLoss: 0.2991

📋 Classification Report:

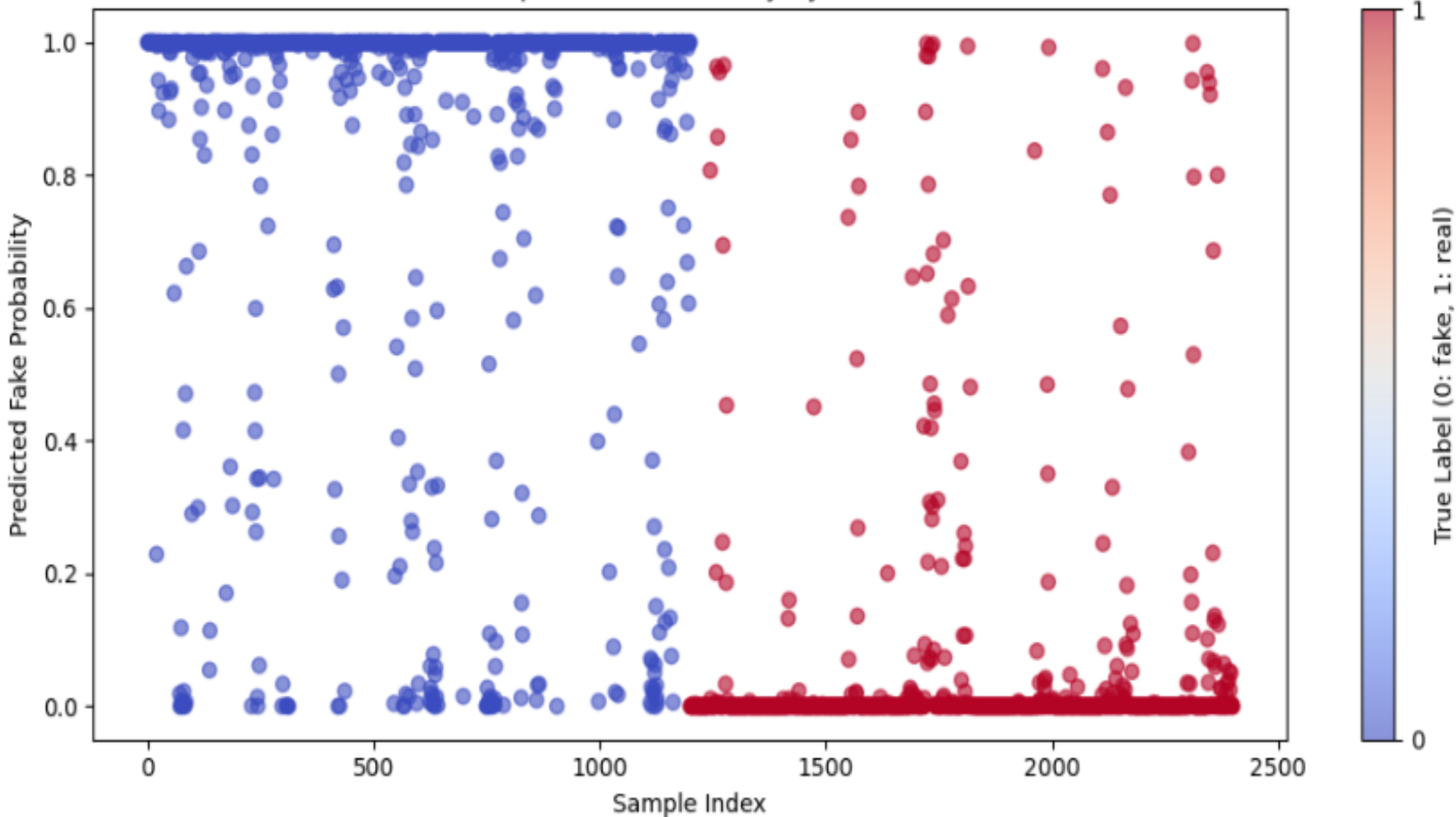
	precision	recall	f1-score	support
fake	0.9620	0.8850	0.9219	1200
real	0.8935	0.9650	0.9279	1200
accuracy			0.9250	2400
macro avg	0.9277	0.9250	0.9249	2400
weighted avg	0.9277	0.9250	0.9249	2400

## 두 번째 데이터 테스트 결과

같은 도메인 내 모델이 학습하지  
않은 데이터셋으로 테스트한 결과



Per-sample Fake Probability by True Label



## 두 번째 데이터 테스트 결과

실제 소프트맥스 결과 값의 그래프



## 앱 / 웹에 들어갈 파일(코드).py

---

```
# 딥페이크 탐지 함수
def detect_deepfake(image_path, efficientnet_weights):
    # 이미지 로드
    image = cv2.imread(image_path)
    if image is None:
        print("이미지를 로드할 수 없습니다.")
        return

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = face_detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

    if len(faces) == 0:
        print("죄송합니다. 얼굴을 탐지할 수 없습니다.")
        return

    print(f"{len(faces)}개의 얼굴을 탐지했습니다.")
```

→ 받은 이미지를 openCV를 통해 얼굴을 탐지한다.



## 앱 / 웹에 들어갈 파일(코드).py

---

```
# 이미지 전처리 함수
def preprocess_image(image):
    preprocess = transforms.Compose([
        transforms.ToPILImage(),
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
    ])
    return preprocess(image).unsqueeze(0).to(device)
```

- 모델에 적합한 이미지 규격을 위해 전처리 진행한다
- 정규화 및 224 x 224 픽셀



## 앱 / 웹에 들어갈 파일(코드).py

```
# EfficientNet 모델 로드
eff_model = create_model("efficientnet")
eff_model.load_state_dict(torch.load(efficientnet_weights, map_location=device))
eff_model.eval()

# 얼굴 탐지 후 각 얼굴 영역에 대해 처리
for i, (x, y, w, h) in enumerate(faces):
    margin = 10
    x1, y1 = max(0, x - margin), max(0, y - margin)
    x2, y2 = min(image.shape[1], x + w + margin), min(image.shape[0], y + h + margin)
    face = image[y1:y2, x1:x2]

    face_tensor = preprocess_image(face)

    # EfficientNet 결과
    with torch.no_grad():
        eff_output = torch.sigmoid(eff_model(face_tensor)).item()
    eff_prediction = "FAKE" if eff_output > 0.5 else "REAL"

    # 결과 출력
    print(f"Face {i+1}: EfficientNet - {eff_prediction} ({eff_output:.4f})")

# 시각화
label = f"E: {eff_prediction}"
color = (0, 0, 255) if eff_prediction == "FAKE" else (0, 255, 0)
cv2.rectangle(image, (x1, y1), (x2, y2), color, 2)
cv2.putText(image, label, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
```

→ 사전 학습한 모델 파일을 통해  
예측을 생성한다



## 출력1

---

1개의 얼굴을 탐지했습니다.

Downloading: "[https://download.pytorch.org/models/efficientnet\\_b0\\_rwrightman-7](https://download.pytorch.org/models/efficientnet_b0_rwrightman-7)  
100%|██████████| 20.5M/20.5M [00:00<00:00, 222MB/s]

```
<ipython-input-2-c962a78cf687>:49: FutureWarning: You are using `torch.load`  
    eff_model.load_state_dict(torch.load(efficientnet_weights, map_location=dev  
Face 1: EfficientNet - FAKE (0.9893)
```





## 출력2

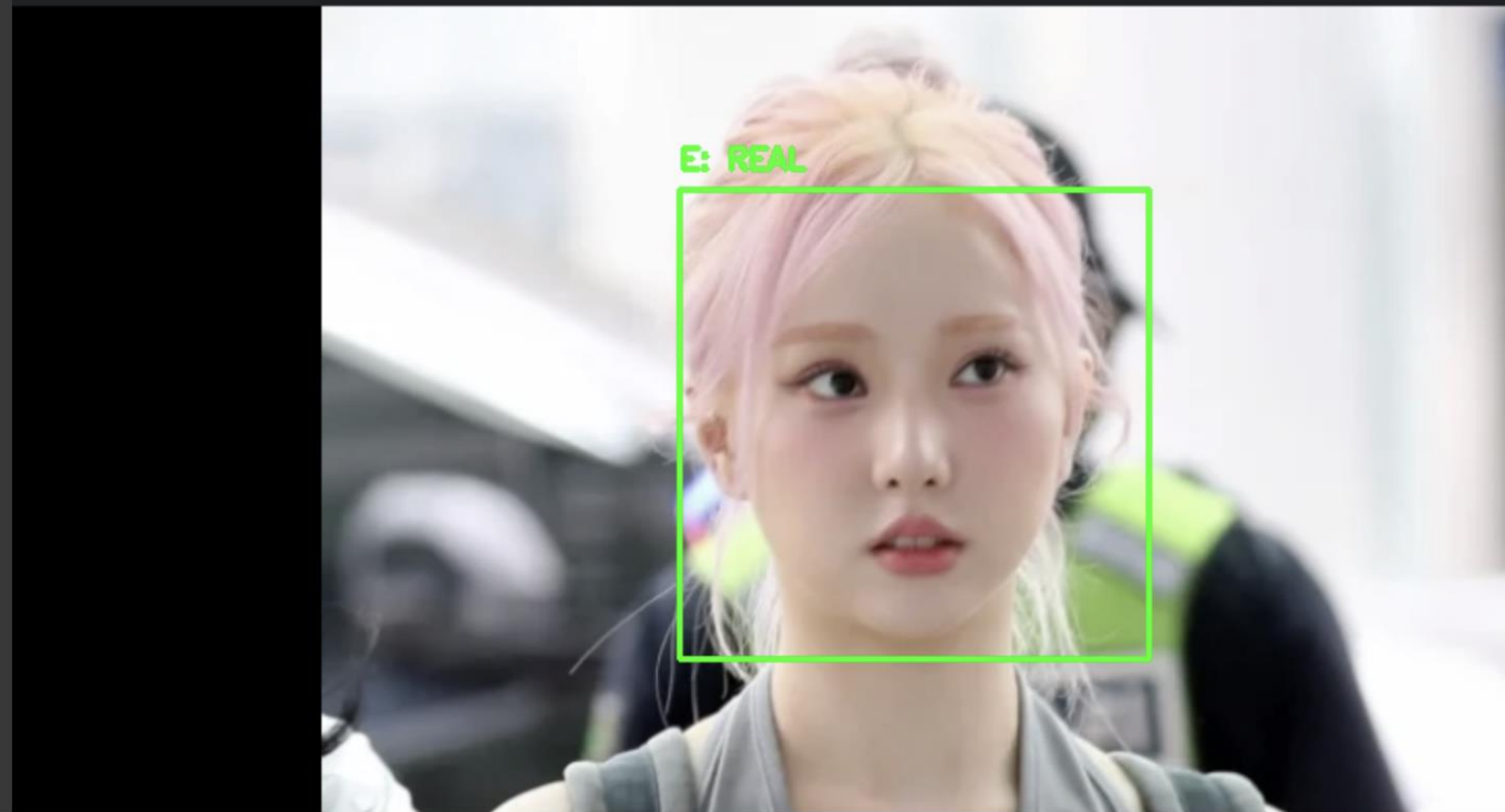
---

1개의 얼굴을 탐지했습니다.

Downloading: "[https://download.pytorch.org/models/efficientnet\\_b0\\_rwrightman-7f5810bc.pth](https://download.pytorch.org/models/efficientnet_b0_rwrightman-7f5810bc.pth)" to /root/.cache/torch/hub/efficientnet\_b0\_rwrightman-7f5810bc.pth  
100%|██████████| 20.5M/20.5M [00:00<00:00, 109MB/s]

```
<ipython-input-1-5010bd75be7a>:49: FutureWarning: You are using `torch.load` with `weights_only=False` which will be deprecated in a future release. Please use `torch.load` with `weights_only=True` to silence this warning. If you are not using weights in the model, the warning can be safely ignored. Recommended safe fix: use `torch.load` with `weights_only=True` (this loading option will be the default in a future release of PyTorch).  
  eff_model.load_state_dict(torch.load(efficientnet_weights, map_location=device))
```

Face 1: EfficientNet - REAL (0.0251)





## 웹 서버 구현

### Deepfake Detection

Start Recording

파일 선택 선택한 파일 없음

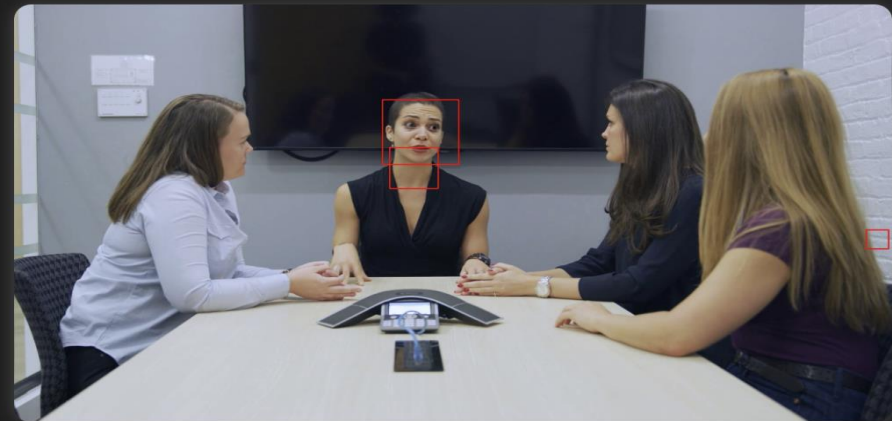
Upload File

### Analysis Result

Fake

Confidence: 0.5011

Detected Face



Go Back