

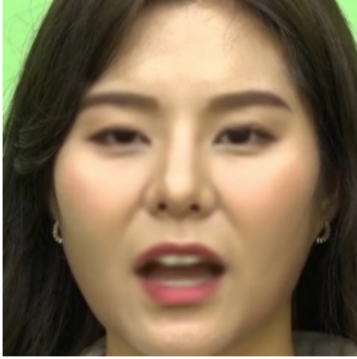
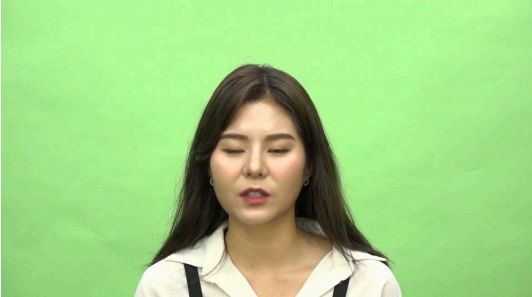







MTCNN 전처리 결과 sample

base	1-a	1-b
		
		
		

## 1-a 코드

```
import os
import cv2
from mtcnn.mtcnn import MTCNN
import matplotlib.pyplot as plt

def crop_faces_in_directory(input_directory, output_directory,
                             target_size=(380, 380), width_margin=0.27, height_margin=0.2,
                             min_size=20, scale_factor=0.709):
    # MTCNN 초기화 (min_size와 scale_factor만 사용)
    detector = MTCNN(min_face_size=min_size, scale_factor=scale_factor)

    # 결과를 저장할 디렉토리가 존재하지 않으면 생성
    if not os.path.exists(output_directory):
        os.makedirs(output_directory)

    # input_directory와 하위 디렉토리 재귀적으로 탐색 모든 JPG 파일 가져옴
    for root, _, files in os.walk(input_directory):
        for filename in files:
            if filename.lower().endswith('.jpg'):
                image_path = os.path.join(root, filename)
                print(f"Processing file: {image_path}")

                # 이미지 읽기
                image = cv2.imread(image_path)
                image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

                # 얼굴 검출
                results = detector.detect_faces(image_rgb)

                # 얼굴 부분을 crop하고 크기 조정
                for i, result in enumerate(results):
                    x, y, w, h = result['box']
                    x, y = max(0, x), max(0, y) #좌표가 음수가 되는 것 방지

                    # 얼굴의 너비와 높이를 가져옴
                    face_width = w
                    face_height = h

                    # 얼굴의 width에 대한 마진 계산
                    width_margin_px = int(face_width * width_margin)
                    x -= width_margin_px
```

```

        face_width += 2 * width_margin_px

        # 얼굴의 height에 대한 마진 계산
        height_margin_px = int(face_height * height_margin)
        y -= height_margin_px
        face_height += 2 * height_margin_px

        # 얼굴 부분 crop
        face = image_rgb[y:y+face_height, x:x+face_width]

        # 얼굴 크기를 target_size로 조정
        face_resized = cv2.resize(face, target_size)

        # 조정된 얼굴 저장
        face_bgr = cv2.cvtColor(face_resized,
cv2.COLOR_RGB2BGR) # 다시 BGR로 변환하여 저장
        output_path = os.path.join(output_directory,
f"{os.path.splitext(filename)[0]}_face_{i+1}.jpg")
        cv2.imwrite(output_path, face_bgr)

        # crop된 얼굴 출력(옵션)
        plt.imshow(face_resized)
        plt.axis('off')
        plt.show()

# 경로 설정
input_directory = '/content/drive/MyDrive/4-1 딥러닝/템플/제목없는
폴더/deepfake_1st/fake/EQ/20200916'
output_directory = '/content/crop2' # 결과 저장 디렉토리 경로

# 함수 사용 예시
crop_faces_in_directory(input_directory, output_directory)

```

## 1-b 코드

```
import os
import cv2
from mtcnn.mtcnn import MTCNN
import numpy as np
import matplotlib.pyplot as plt

def crop_faces_in_directory(input_directory, output_directory,
                             target_size=(380, 380), min_size=20, scale_factor=0.709):
    # MTCNN 초기화 (min_size와 scale_factor만 사용)
    detector = MTCNN(min_face_size=min_size, scale_factor=scale_factor)

    # 결과를 저장할 디렉토리가 존재하지 않으면 생성
    if not os.path.exists(output_directory):
        os.makedirs(output_directory)

    # input_directory와 그 하위 디렉토리를 재귀적으로 탐색하여 모든 JPG 파일을 가져옴
    for root, _, files in os.walk(input_directory):
        for filename in files:
            if filename.lower().endswith('.jpg'):
                image_path = os.path.join(root, filename)
                print(f"Processing file: {image_path}")

                # 이미지 읽기
                image = cv2.imread(image_path)
                image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

                # 얼굴 검출
                results = detector.detect_faces(image_rgb)

                # 얼굴 부문 crop & 크기 조정
                for i, result in enumerate(results):
                    x, y, w, h = result['box']
                    x, y = max(0, x), max(0, y)  #좌표 음수 되는 것 방지

                    # 얼굴 중심점 계산
                    center_x = x + w // 2
                    center_y = y + h // 2

                    # 얼굴 width와 height 중 최댓값 계산
                    face_size = max(w, h)
```

```

        # 새로운 Bounding Box의 좌상단 점 계산
        new_x = max(0, center_x - face_size // 2)
        new_y = max(0, center_y - face_size // 2)

        # 새로운 Bounding Box의 우하단 점 계산
        new_w = min(image.shape[1], center_x + face_size //
2)

        new_h = min(image.shape[0], center_y + face_size //
2)

        # 얼굴 부분 crop
        face = image_rgb[new_y:new_h, new_x:new_w]

        # 얼굴을 target_size로 resize
        face_resized = cv2.resize(face, target_size)

        # zero padding을 실시하여 target_size에 맞게 조정
        padded_face = np.zeros((target_size[1],
target_size[0], 3), dtype=np.uint8)
        padded_face[:face_resized.shape[0],
:face_resized.shape[1], :] = face_resized

        # 조정된 얼굴을 저장
        output_path = os.path.join(output_directory,
f"{os.path.splitext(filename)[0]}_face_{i+1}.jpg")
        cv2.imwrite(output_path, cv2.cvtColor(padded_face,
cv2.COLOR_RGB2BGR))

        # crop된 얼굴을 출력(옵션)
        plt.imshow(padded_face)
        plt.axis('off')
        plt.show()

# 경로 설정
input_directory = '/content/drive/MyDrive/4-1 딥러닝/템플/제목없는
폴더/deepfake_1st/fake/EQ/20200916'
output_directory = '/content/crop3' # 결과저장 디렉토리 경로

# 함수 사용 예시
crop_faces_in_directory(input_directory, output_directory)

```