



# 얼굴 유사도 검사

## face\_recognition

라이브러리 = 딥러닝 기반 AI 모델

기반: dlib 라이브러리

딥러닝 모델: ResNet-34 (ResNet 기반 Face Recognition 모델)

학습 데이터: 얼굴 인식용으로 수십만 장의 얼굴 이미지로 학습된 **Deep Metric Learning** 모델

1. **얼굴을 감지** → CNN 기반 HOG + CNN 얼굴 감지기 사용
2. **얼굴을 인코딩 (Embedding)** → 128차원 벡터로 변환
3. **벡터 간 거리 계산** → 두 이미지의 얼굴 벡터 간 거리(Euclidean Distance)

```
import face_recognition
import cv2

def compare_faces(id_image_path, selfie_image_path):
    # 신분증 이미지 로드
    id_image = face_recognition.load_image_file(id_image_path)
    id_encodings = face_recognition.face_encodings(id_image)

    # 셀카 이미지 로드
    selfie_image = face_recognition.load_image_file(selfie_image_path)
    selfie_encodings = face_recognition.face_encodings(selfie_image)

    if len(id_encodings) == 0 or len(selfie_encodings) == 0:
        return {"match": False, "similarity": 0.0, "error": "Face not detected in one or both images"}

    id_encoding = id_encodings[0]
    selfie_encoding = selfie_encodings[0]

    # 거리 계산 (거리 작을수록 비슷)
```

```

distance = face_recognition.face_distance([id_encoding], selfie_encoding)[0]
similarity = 1 - distance # 1에 가까울수록 유사함

threshold = 0.6 # 일반적으로 0.6 이하이면 동일 인물로 간주
is_match = distance < threshold

return {
    "match": is_match,
    "similarity": similarity,
    "distance": distance
}

# 사용 예시
result = compare_faces("id_photo.jpg", "selfie.jpg")
print(result)

```

일단 하는 방법은 이런식이 가장 적합한 것 같은데 어떤걸더 조사를 해야할지 잘 모르겠어요  
ㅌㅌ






직접 할때

## 1. 데이터 준비

- 신분증 사진 + 셀카 쌍 (동일인/다른인)
  - 이걸 모델이 알아서 함 얼굴뻥 되있는거 하면 됨

## 2. 모델 구조 선택

- ArcFace, FaceNet, VGGFace 등 얼굴 인식용 딥러닝 모델

항목	ArcFace	FaceNet	VGGFace
 개발자	MS Asia	Google	VGG (Oxford)
 구조	ResNet + Additive Angular Margin	Inception + Triplet Loss	VGGNet
 목적	얼굴 인증 (정확도 ↑)	얼굴 유사도 학습	얼굴 분류
 정확도 (LFW 기준)	<b>99.83%</b>	99.63%	98.95%
 학습 방식	CosFace + Angular Margin (classification)	Triplet Loss	Softmax Classification

⚙️ 활용 용도	인증, 유사도, 산업적 사용	유사도 계산	분류, 간단한 인식
💡 장점	가장 정확도 높음, 산업표준	구현 간단, 학습 빠름	쉬움, 구조 단순
❗ 단점	학습이 복잡, 리소스 요구	Triplet 구성 까다로 움	정확도 낮음, 구식 모 델

### 🔧 3. 학습

- Triplet loss 또는 Cosine similarity 기반 학습

항목	Triplet Loss	Cosine Similarity Loss
🌱 개념	Anchor, Positive, Negative 세 개 샘플의 거리 조정	같은 클래스는 코사인 유사도 ↑, 다른 클래스는 ↓
🔧 수식	$\max(0, d(a,p) - d(a,n) + \text{margin})$	$1 - \cos(\theta)$
🍲 학습 난이도	어려움 (triplet mining 필요)	간단함
📈 성능	잘 튜닝하면 강력함	일반적으로 안정적
✅ 장점	좋은 유사도 학습 가능	학습 안정성 높음
❌ 단점	triplet 샘플 구성 어려움	성능 한계 존재

ArcFace는 **Cosine-based Margin Loss**

PyTorch 또는 TensorFlow 사용

특히 얼굴 인식 모델 커스터마이징하고 실험 많이 할 거라면 PyTorch가 훨씬 **쉽고 유연**합니다.