

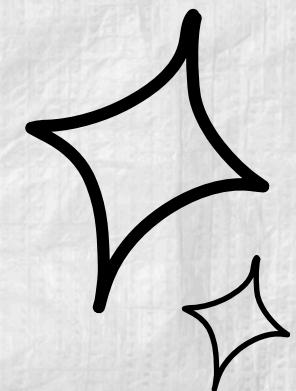
종 분류 기반 개체 식별을 통한

야생동물 재식별



우~차

1. 분류와 재식별
2. AnimalCLEF 2025 대회 소개
3. 시행착오
4. Pipeline
5. 결과 분석 및 해설
6. 활용방안



팀원 소개



동연



한준



수자



채연



보희

1

분류(classification)과 재식별(Re-identification)



분류(Classification)와 재식별(Re-identification)

1. 분류(Classification)

- 정의: 미리 정의된 종류(예: 고양이, 개 등) 중 어떤 그룹에 속하는지 분류
- 특징: 클래스 수 고정, 같은 종 내 개체 구분 X
- 예시: "이건 고양이야!"

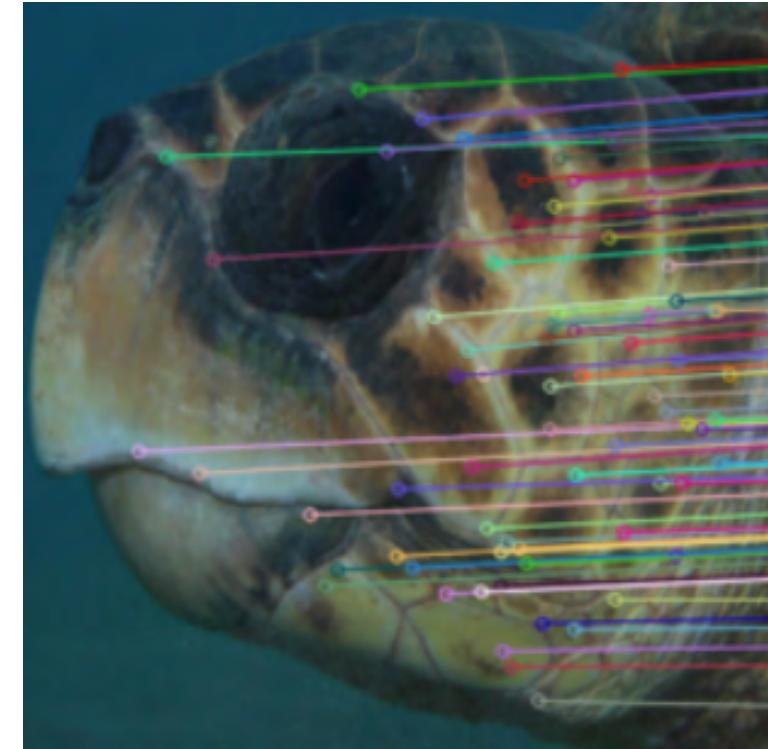
2. 재식별(Re-identification)

- 정의: 이 개체가 전에 본 그 개체와 같은지 확인
- 특징: 개체 수 무한, 처음 보는 개체와도 비교, 외형 변화(옷, 각도 등) 있어도 식별 필요
- 예시: "이 고양이는 어제 본 고양이 A야!"

왜 재식별이 더 어려운가?

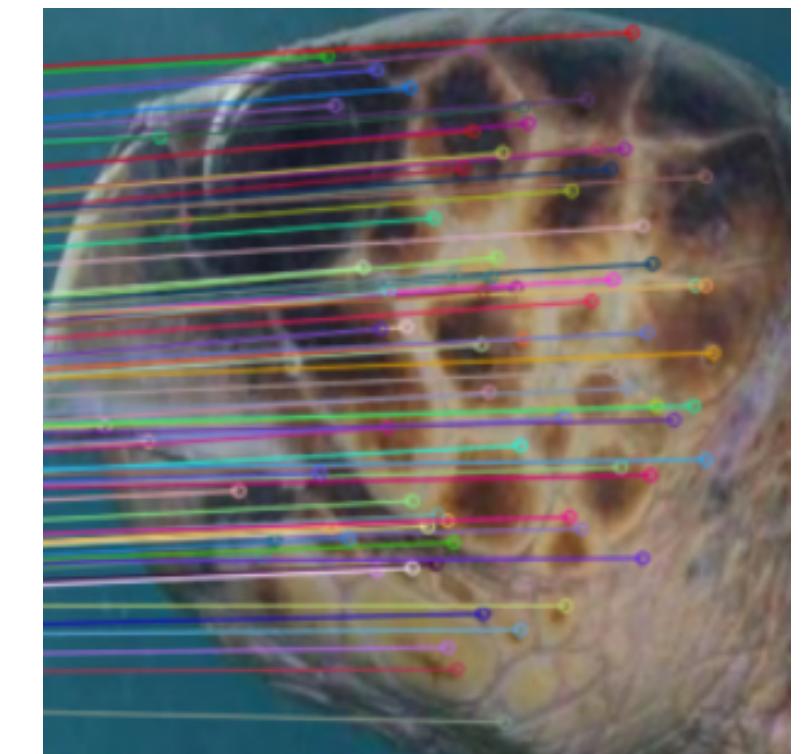
1. 처음 보는 개체 등장

→ 학습에 없던 개체도 식별해야 함



2. 외형 변화

→ 각도, 조명, 포즈, 옷 등이 달라도 같은 개체로 인식해야 함

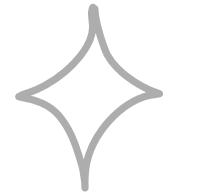


3. 데이터 부족

→ 개체당 사진이 1~2장뿐인 경우도 많음 (one-shot 문제)

2

AnimalCLEF 2025 대회 소개



AnimalCLEF 2025 대회 소개



- AnimalCLEF 2025 대회는 CVPR 2025의 FGVC (Fine-Grained Visual Categorization) 워크숍과도 연계
- CVPR은 IEEE가 주최하는 컴퓨터 비전 및 패턴 인식 분야에서 가장 권위 있는 국제 학술 대회



AnimalCLEF 2025 대회 소개

1. 목표

특정 동물(개체)의 이미지를 보고

- 처음 보는 개체인지,
- 이미 본 적 있는 개체라면 누구인지를 맞히는 것.

2. 대상 종

- 바다거북 (Greece)
- 도롱뇽 (Czech Republic)
- 시라소니 (Czech Republic)

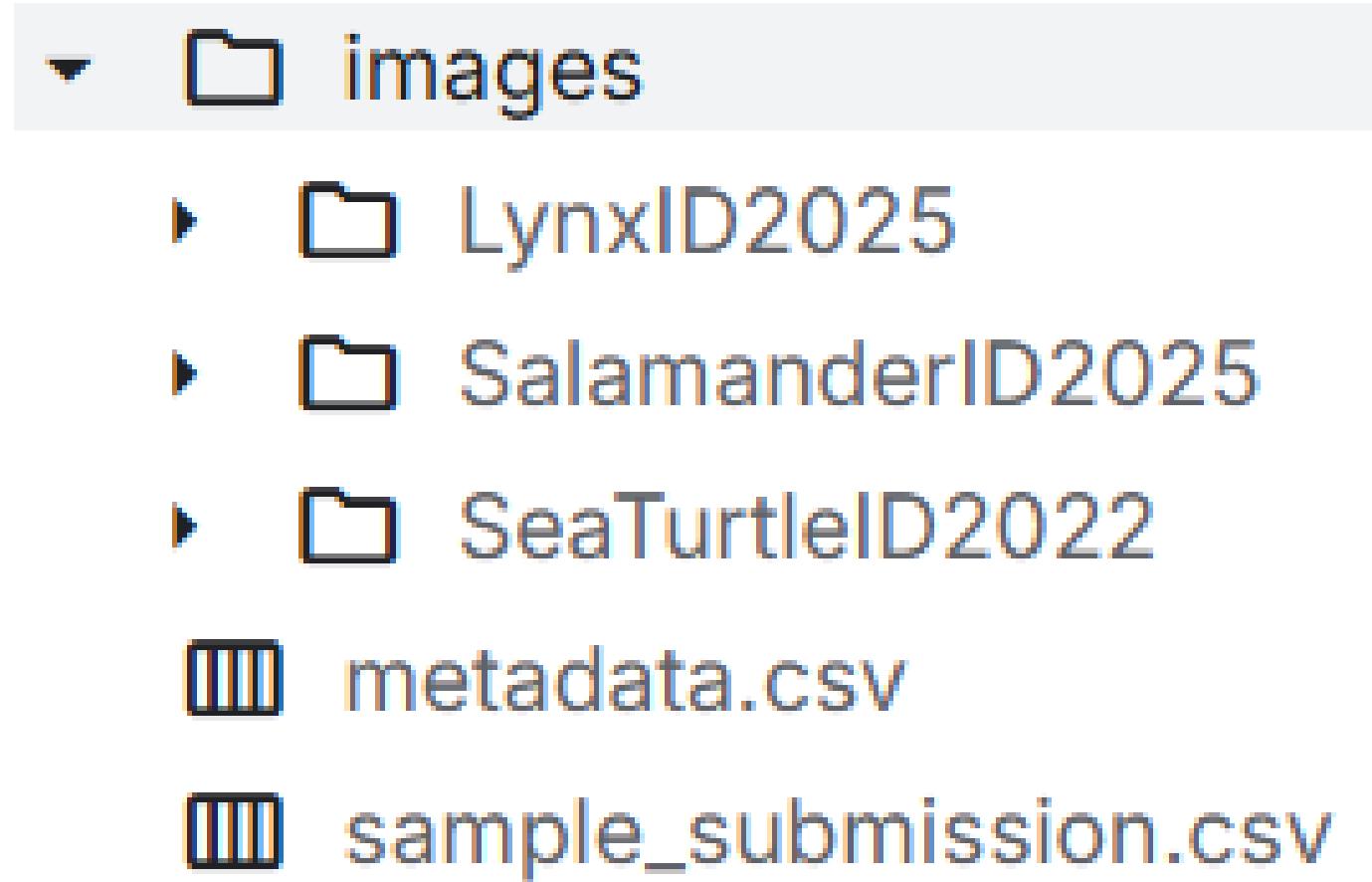
3. 도전 과제

- 재식별 기반 문제: 종 분류가 아닌 개체 단위 식별
- 훈련에 없는 개체 등장 가능
- 조명/각도/포즈 변화 극복 필요

4. 평가기준

- known 개체의 정확도 (BAKS)
- unknown 개체의 정확도 (BAUS)
- → 두 정확도의 기하 평균을 최종 점수로 사용

데이터셋



데이터 구조

- Database

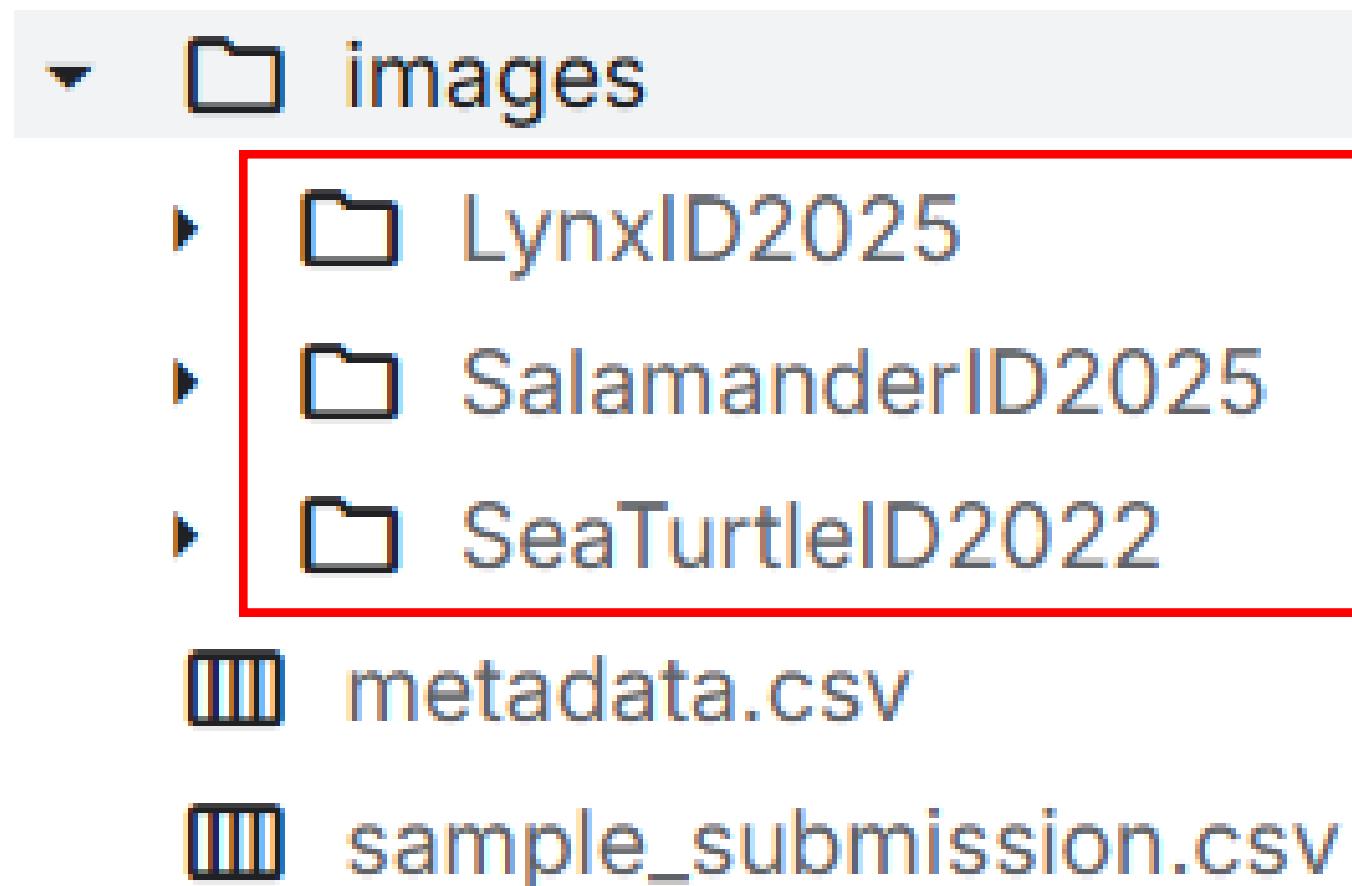
정답이 있는 DB셋으로 각 개체 ID가 라벨로 주어짐
(예: LynxID2025_lynx_17)

- Query

찾고자 하는 이미지셋으로 각 이미지가 DB에 있는지
(new or known) 판단해야 함

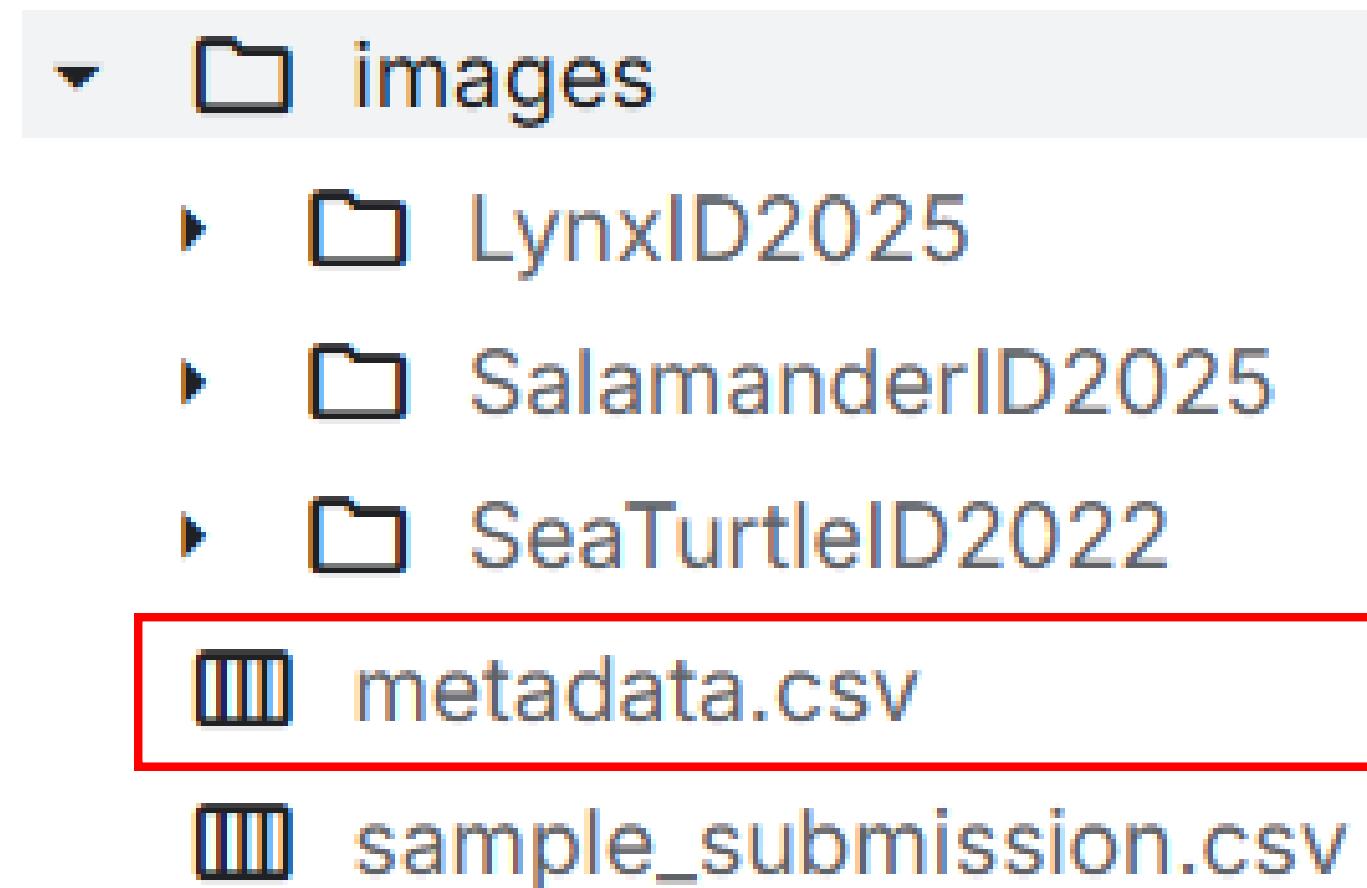
데이터셋

동물 이미지 데이터



데이터셋

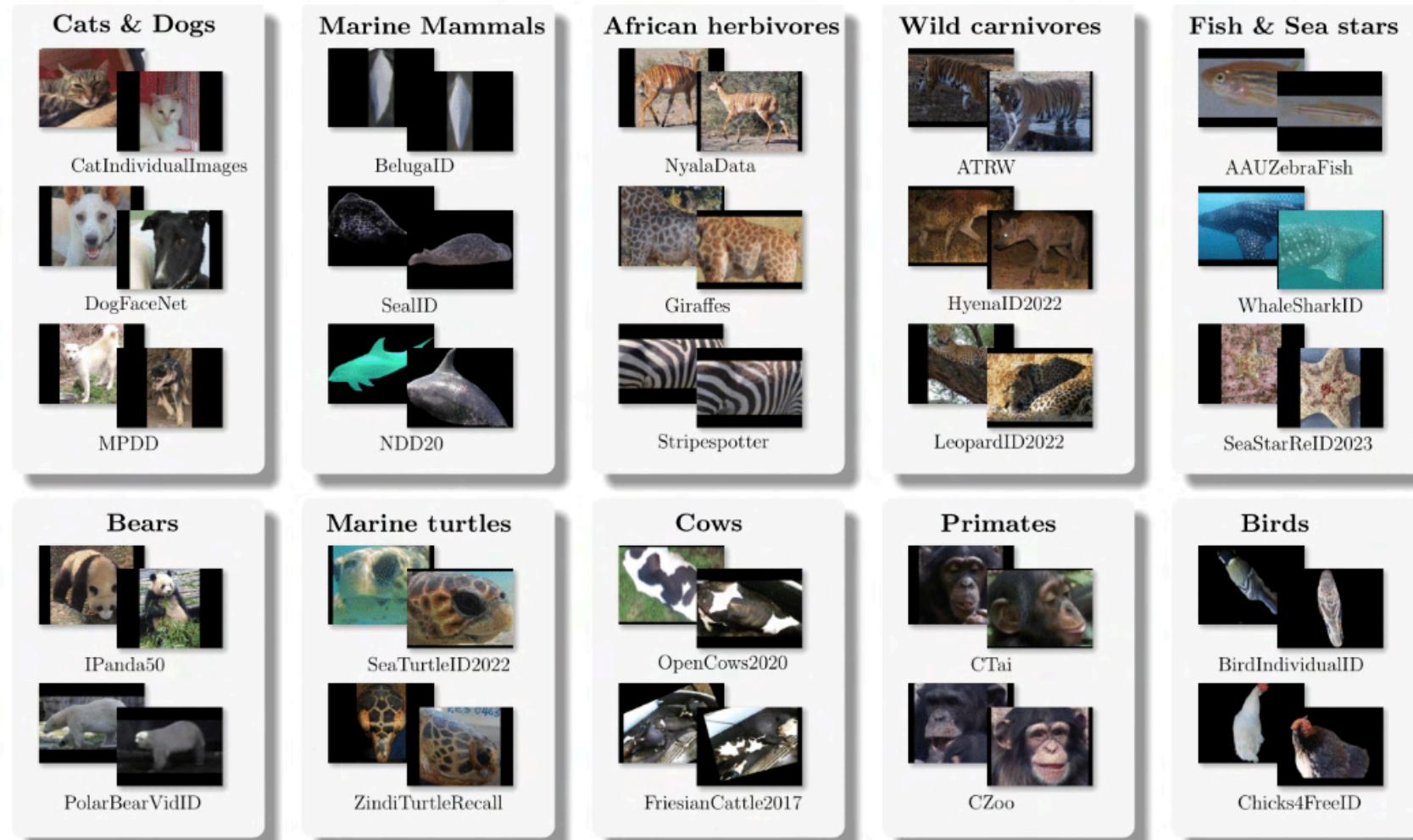
metadata.csv



- 이미지 경로, ID, 종, 날짜, 방향 정보 등 포함

	A	B	C	D	E	F	G	H
1	image_id	identity	path	date	orientationspecies	split	dataset	
2	0	LynxID202	images/LynxID2025/cright		lynx	database	LynxID2025	
3	1	LynxID202	images/LynxID2025/cleft		lynx	database	LynxID2025	
4	2	LynxID202	images/LynxID2025/cleft		lynx	database	LynxID2025	
5	3		images/LynxID2025/cback		lynx	query	LynxID2025	
6	4	LynxID202	images/LynxID2025/cright		lynx	database	LynxID2025	
7	5		images/LynxID2025/cleft		lynx	query	LynxID2025	
8	6	LynxID202	images/LynxID2025/cleft		lynx	database	LynxID2025	
9	7	LynxID202	images/LynxID2025/cleft		lynx	database	LynxID2025	
10	8	LynxID202	images/LynxID2025/cfront		lynx	database	LynxID2025	
11	9	LynxID202	images/LynxID2025/cright		lynx	database	LynxID2025	
12	10	LynxID202	images/LynxID2025/cright		lynx	database	LynxID2025	
13	11	LynxID202	images/LynxID2025/cfront		lynx	database	LynxID2025	
14	12		images/LynxID2025/cleft		lynx	query	LynxID2025	
15	13		images/LynxID2025/cright		lynx	query	LynxID2025	
16	14	LynxID202	images/LynxID2025/cleft		lynx	database	LynxID2025	
17	15	LynxID202	images/LynxID2025/cright		lynx	database	LynxID2025	

학습용 데이터셋(선택)



WildlifeReID-10k

- 14만 장 이미지, 1만 개체 이상
- 다양한 종 포함 (거북, 유인원, 조류 등)
- 모델 사전학습(Pre-training)용으로 활용 가능

3 시행착오



[paper] A ConvNet for the 2020s

A ConvNet for the 2020s

Zhuang Liu^{1,2*} Hanzi Mao¹ Chao-Yuan Wu¹ Christoph Feichtenhofer¹ Trevor Darrell² Saining Xie^{1†}

¹Facebook AI Research (FAIR) ²UC Berkeley

Code: <https://github.com/facebookresearch/ConvNeXt>

ConvNeXt 특징

- ResNet 기반
- CNN 구조를 유지하면서도 Transformer보다 더 나은 정확도
- 학습 안정성 향상 : LN, 적절한 downsampling 설계
- 연산 효율성 : Depthwise conv + Inverted Bottleneck으로 FLOPs 대비 성능 향상

❖ 코드 실행을 통해 파악한 ConvNeXt 문제점

```
87 class ConvNeXtClassifier(nn.Module):
88     def __init__(self, model_name='convnext_tiny', num_classes=None):
89         super().__init__()
90         self.backbone = timm.create_model(model_name, pretrained=False, num_classes=num_classes)
91
92     def forward(self, x):
93         return self.backbone(x)
```

- (1) wildlife dataset으로 처음부터 사전학습 시킨 뒤 가중치 저장
- (2) metadata.csv의 dataset의 열 정보로 이에 맞는 종에서 소분류 진행

ConvNeXt 문제점

- wildlife 데이터셋이 너무 커, 사전 학습에 많은 시간 소요됨
- CNN 계열은 long-range dependency 학습에 Transformer 구조보다 불리함
- 고해상도 기반 학습에 최적화되어 있음.
** 재식별(Re-ID)은 종종 저해상도나 부분 occlusion 상황에서 수행됨.



Dataset를 통해 본 분류와 재식별 방법의 차이



시라소리(LynxID2025)



도롱뇽(SalamanderID2025)



바다거북(SeaTurtleID2022)

- 1) 클래스 수가 고정되어 있지 않음
- 2) 이전에 식별한 적 없는 개체에 대한 식별도 필요
=> 특징 벡터를 계산해서 특징 벡터 간 거리(임베딩 거리)로 식별해야 함

❖ 해결 방향성

WildlifeDatasets: An open-source toolkit for animal re-identification

Vojtěch Čermák¹, Lukas Picek^{2,3}, Lukáš Adam¹ and Kostas Papafitsoros⁴
¹Czech Technical University in Prague, ²University of West Bohemia, ³INRIA
⁴Queen Mary University of London

=> 사전학습 된 모델(MegaDescriptor)
사용해 고품질의 특징 벡터 추출

- Swin Transformer 기반 구조로, global과 local 특징 파악에 용이
→ **다양한 해상도의 이미지를 처리**하는 데 유리 + **복잡한 배경이나 다양한 포즈**의 동물 이미지를 효과적으로 분석
- ArcFace 손실 함수를 사용하여 학습하여, 각 클래스 간의 각도 차이를 최대화하여 더 구별력 있는 임베딩을 생성
→ **유사한 개체 간의 구분**이 향상됨
- 29개의 공개 동물 재식별 데이터셋을 활용하여 학습되어, **다양한 종에 대한 일반화된 특징**을 학습함

따라서, **MegaDescriptor**는 재식별 모델의 백본으로 선택

✨ Rerank Cascade

파이프라인

- **1차 예측:** MegaDescriptor + EVA02 파이프라인에서 이미지의 유사도 추출
- **2차 예측:** 각 쿼리 이미지에 대해 가장 유사한 Tok-K 후보 이미지를 골라 그 쿼리와 DB 이미지를 ALIKED local Matcher로 재비교
- **기대효과:**
 - 각 쿼리 이미지에 대해 k개의 후보를 비교해 정확도 향상
 - k개의 후보에 대해서만 local matcher를 수행해 속도 향상



오히려 정확도가 낮아지는 결과
LoFTR matcher 도입 시도



sample_submission.csv

Complete · dongyeonkim10 · 1mo ago · Rerank Cascade (cutoff 0.35)

0.42590

LoFTR matcher

- ALIKED 대신 Dense matching 방식의 local feature matcher인 LoFTR 도입 시도

Local Matcher

구분	Keypoint 기반	Dense Matching 기반
매칭 방식	중요한 지점에 대해서만 매칭	전체 이미지에 대해서 매칭
예시	ALIKED	LoFTR



ALIKED와 유사한 결과
Top-K 후보가 이미 너무 정확해
Rerank 방식의 정확도가 더 낮은
것으로 해석



sample_submission (1).csv
 Complete · dongyeonkim10 · 1mo ago · LoFTR matcher 도입

0.41803

❖ Fusion MLP

배경

- 기존 코드는 MegaDescriptor와 EVA02 모델의 feature를 정적인 가중치로 결합

```
combined_sim = 0.5 * sim_fusion + 0.5 * sim_eva
```



- 두 피쳐 사이의 비선형적 관계를 학습하는 MLP 모델을 뒤 해당 모델이 피쳐를 결합해 최종 유사도 스코어 출력

- MLP를 학습시키지 않고 랜덤 초기화 상태로 사용했을 땐 더 낮은 정확도를 보임
- MLP 학습에 큰 리소스가 필요해 학습을 완료하지 못함



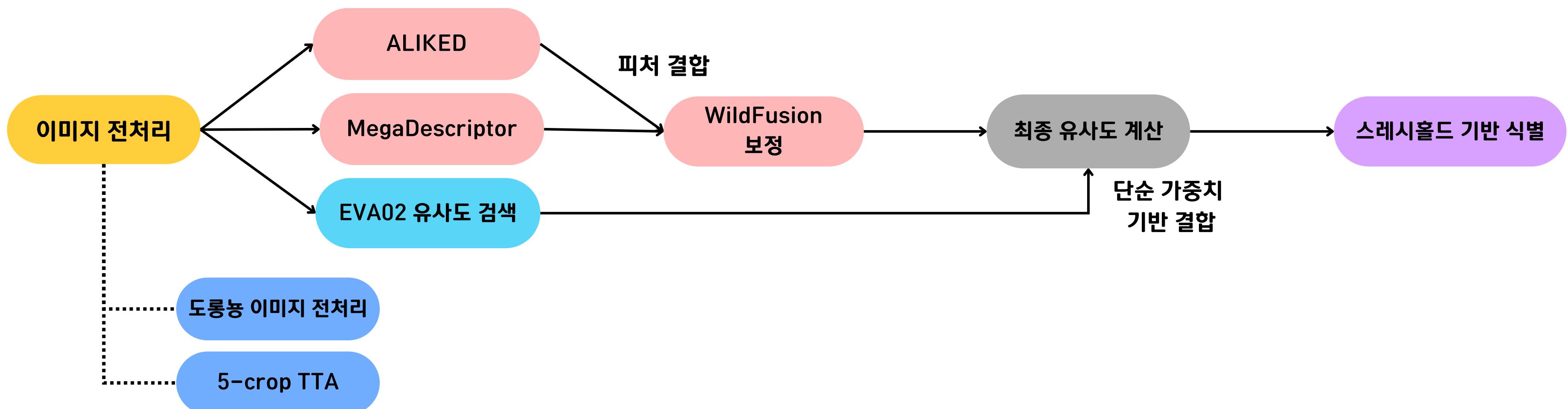
sample_submission (1).csv

Complete · dongyeonkim10 · 10d ago · fusion MLP 랜덤 가중치 테스트

0.49906

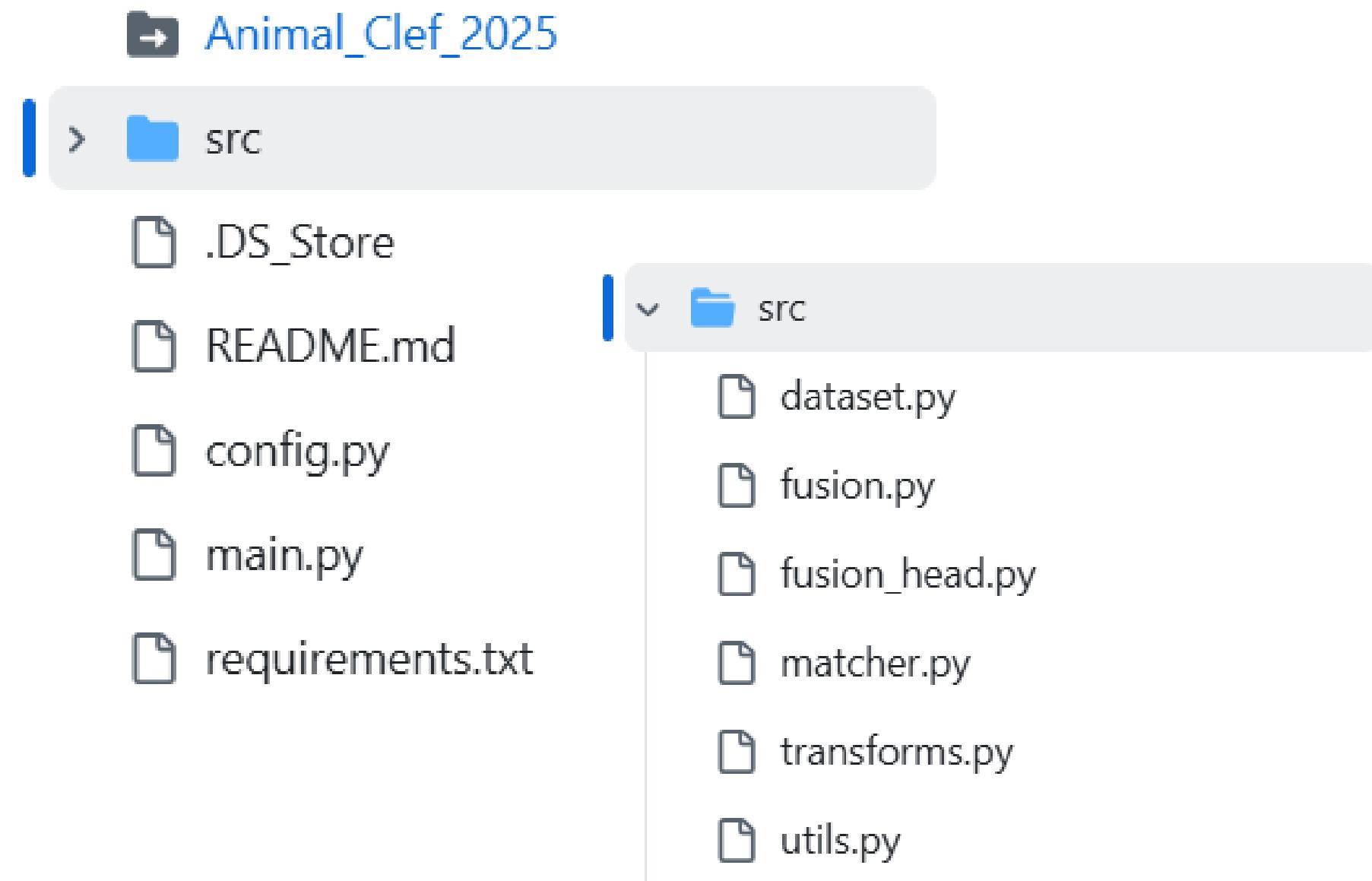
4 Pipeline

_pipeline



★ 주요 코드

https://github.com/dongyeon1031/bitamin_cv_proj



main.py

- 전체 파이프라인 실행 스크립트

dataset.py

- 데이터셋 로딩 및 전처리 함수 정의

matcher.py

- 각 matcher를 구성하는 로직 정의

fusion.py

- WildFusion 구성 및 calibration 적용 모듈

transforms.py

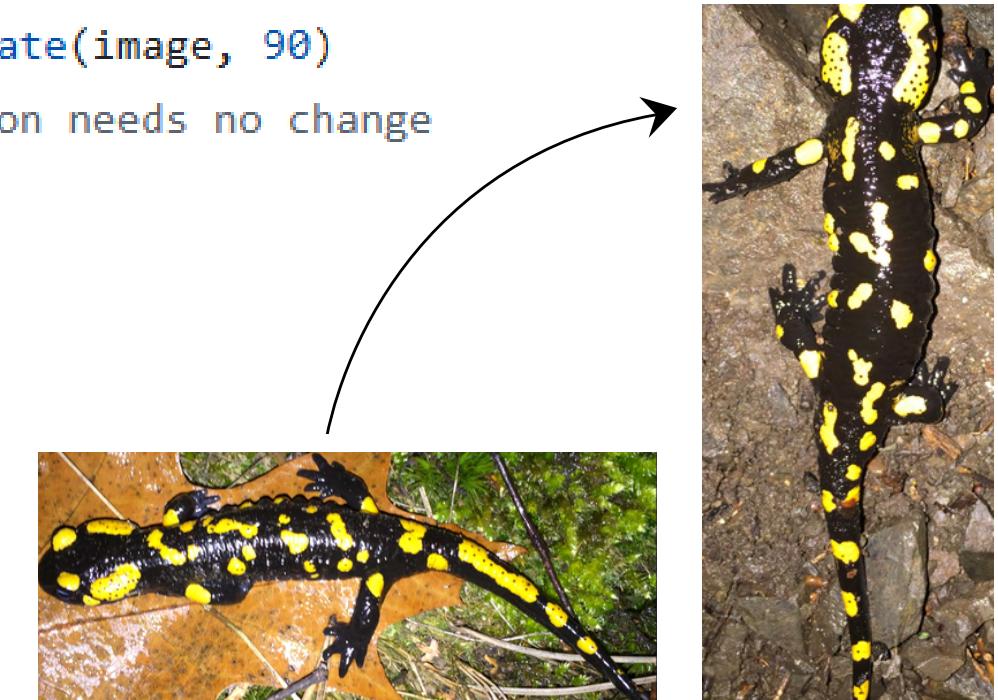
- 이미지 전처리 정의 파일

◆ 도록농 이미지 전처리

도록농 이미지의 방향을 일관성 있게 맞춤

- 메타데이터의 orientation 값 활용
- orientation 값이 right이면 이미지를 -90도 회전
orientation 값이 left이면 이미지를 +90도 회전

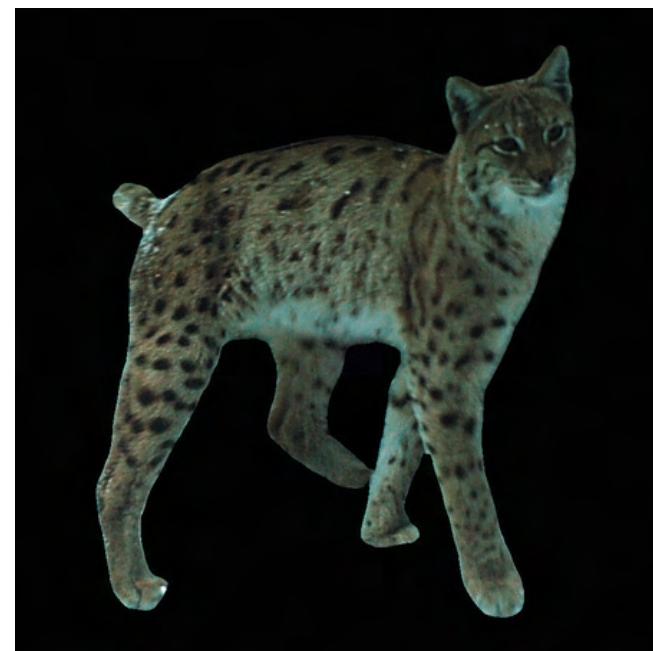
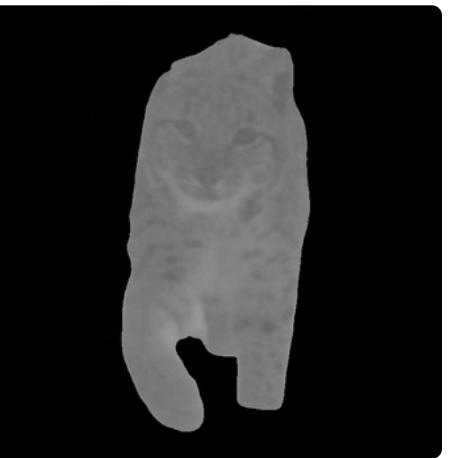
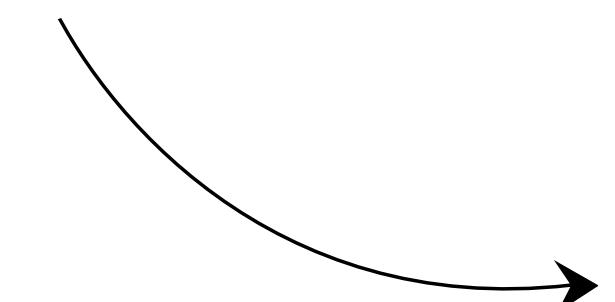
```
def salamander_orientation_transform(image, metadata):  
    # Only apply to SalamanderID2025 dataset  
    if metadata.get("dataset") == "SalamanderID2025":  
        orientation = metadata.get("orientation", "top")  
        # Align 'right' orientation to 'top' by rotating -90 degrees  
        if orientation == "right":  
            return TF.rotate(image, -90)  
        # Align 'left' orientation to 'top' by rotating +90 degrees  
        elif orientation == "left":  
            return TF.rotate(image, 90)  
        # 'top' orientation needs no change  
    return image
```



❖ 5-crop TTA

TTA

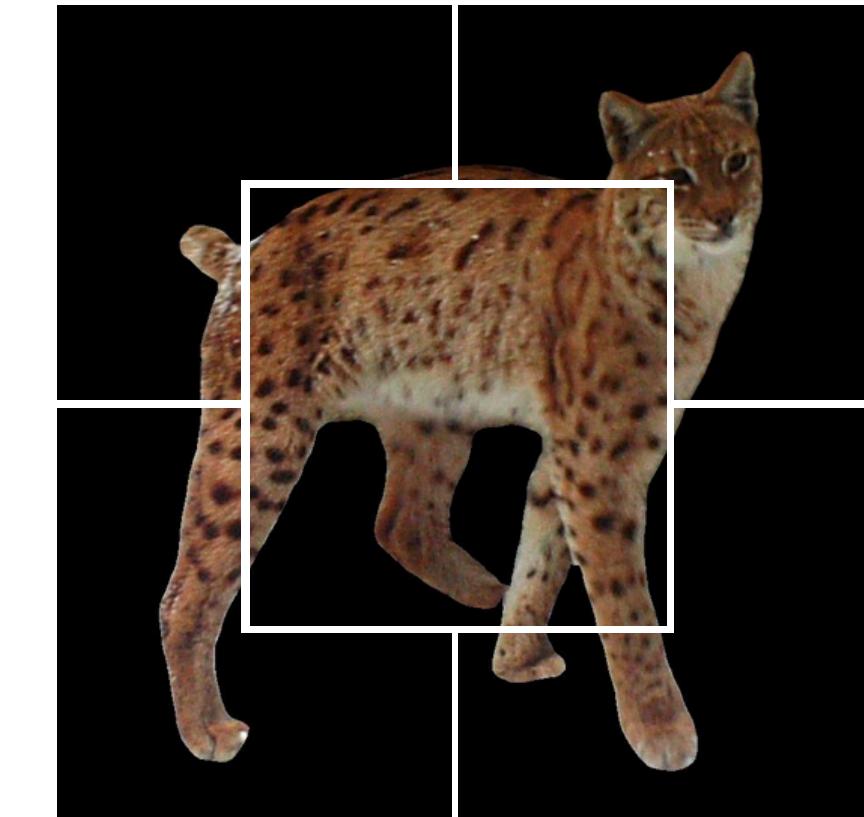
- 테스트 단계에서 입력 이미지를 여러 방식으로 변형하여 여러 번 예측을 수행하고 그 결과를 평균 내 성능을 높이는 기법
- 야간 이미지가 많은 시라소니 분류에 큰 효과를 기대할 수 있음



❖ 5-crop TTA

```
# ----- Five-Crop TTA (average) helper -----
class FiveCropAverage:
    ...
    Apply torchvision FiveCrop then average the 5 cropped tensors into one.
    Output tensor shape == single image tensor; thus DeepFeatures works unchanged.
    ...
    def __init__(self, size, mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)):
        self.five_crop = T.FiveCrop(size)
        self.to_tensor = T.ToTensor()
        self.normalize = T.Normalize(mean=mean, std=std)

    def __call__(self, img):
        crops = self.five_crop(img)          # tuple of 5 PIL images
        tensors = [self.normalize(self.to_tensor(c)) for c in crops]
        stacked = torch.stack(tensors, dim=0) # 5 x C x H x W
        return stacked.mean(dim=0)          # C x H x W (averaged)
```



이미지를 5조각으로 나눠 부분별 예측을 한
뒤 평균을 내 최종 예측을 수행하는 방법

4. Pipeline

ALIKED Fusion

ALIKED

- [paper] 2023년 IEEE에서 소개

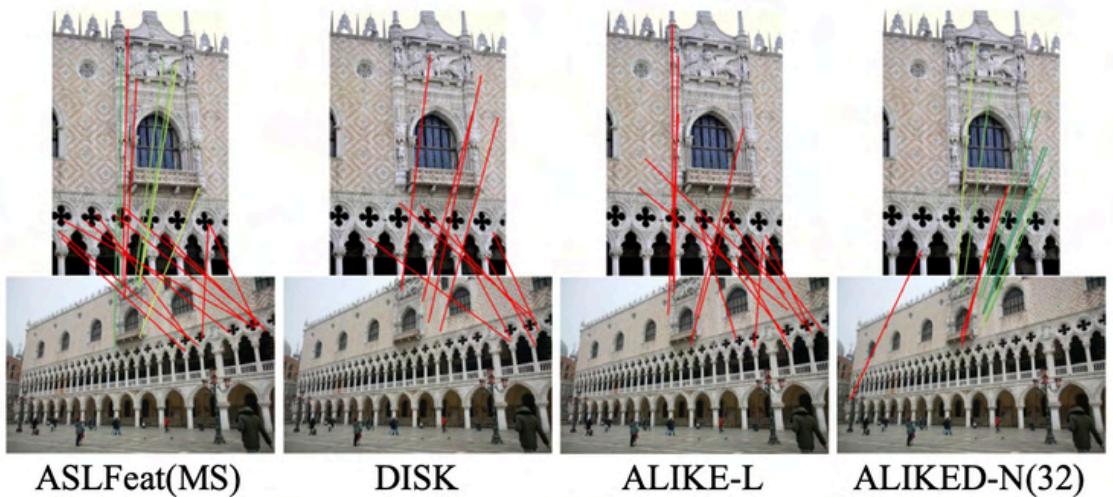


Fig. 8: Matching results of SOTA methods for images with large differences in scale and viewpoint. The matches are colored from green to yellow based on their reprojection error, which ranges from 0 to 5 pixels. False matches are marked in red. Best viewed in color and zoomed in at 400%.

MegaDescriptor

- [paper] 2023년 CVPR에서 소개

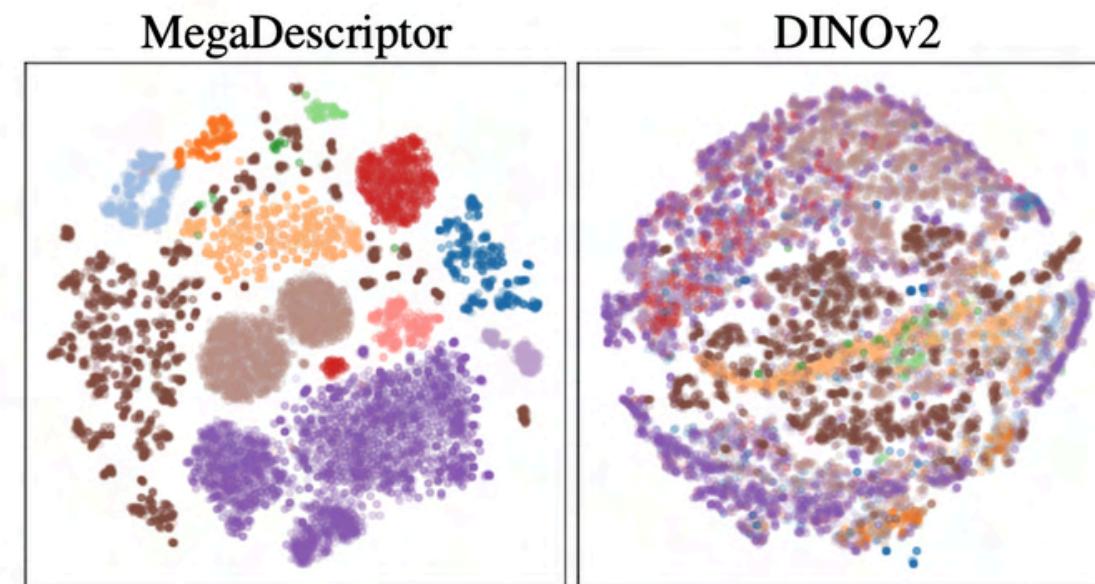
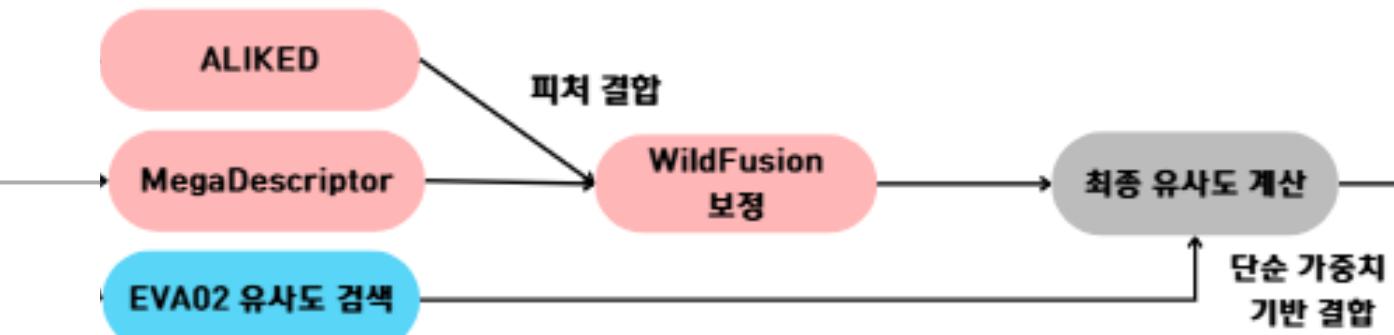
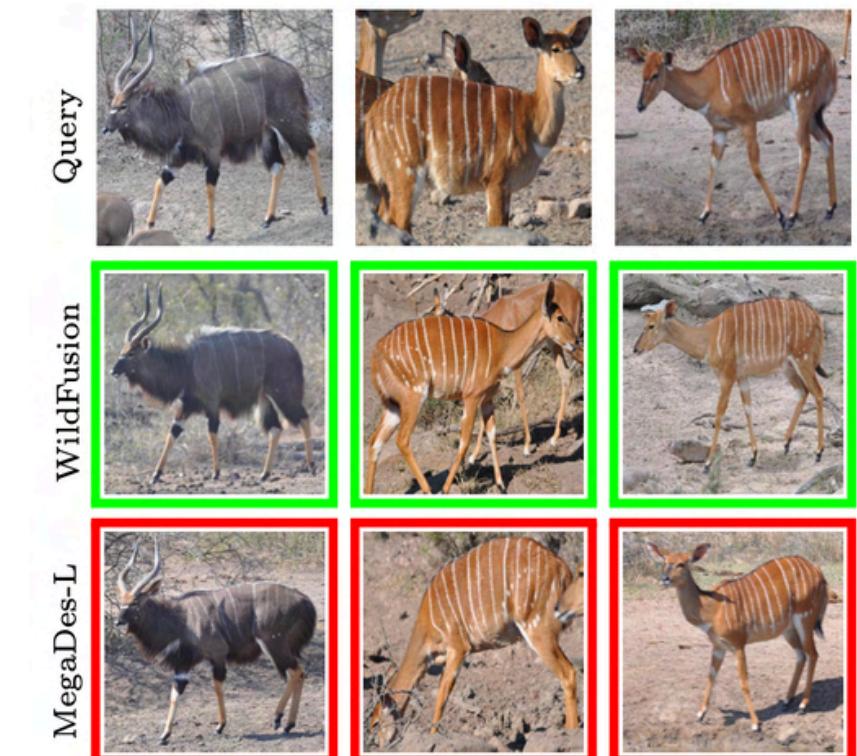


Figure 1. **Latent space separability of MegaDescriptor.** Embedding visualization (t-sne) of unseen individual animals (identity-wise) for the proposed MegaDescriptor and DINOv2. Colors represent different datasets (i.e., species).



WildFusion

- [paper] 2024년 프리프린트

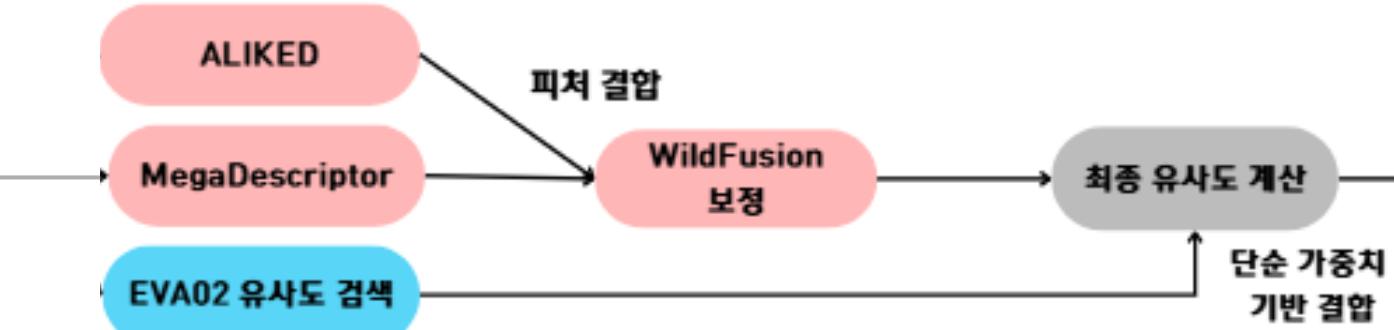


4. Pipeline

◆ ALIKED Fusion

여러 백본에서 추출된 특징 벡터를 결합하여 최종 이미지 특징을 구성

- 각 모델의 강점을 하나의 특징에 융합하는 기능
- ALIKED와 MegaDescriptor 모델을 Fusion 후 WildFusion을 통해 보정(calibration) 후 최종 결합



```
...
MegaDescriptor, ALIKED matcher 각각 생성 + return
...
def build_megadescriptor(model, transform, device='cuda', batch_size=16):
    return SimilarityPipeline(
        matcher=CosineSimilarity(),
        extractor=DeepFeatures(model=model, device=device, batch_size=batch_size),
        transform=transform,
        calibration=IsotonicCalibration()
    )

# --- ALIKED matcher -----
def build_aliked(transform, device='cuda', batch_size=16):
    return SimilarityPipeline(
        matcher=MatchLightGlue(features='aliked', device=device, batch_size=batch_size),
        extractor=AlikedExtractor(),
        transform=transform,
        calibration=IsotonicCalibration()
    )
```

4. Pipeline

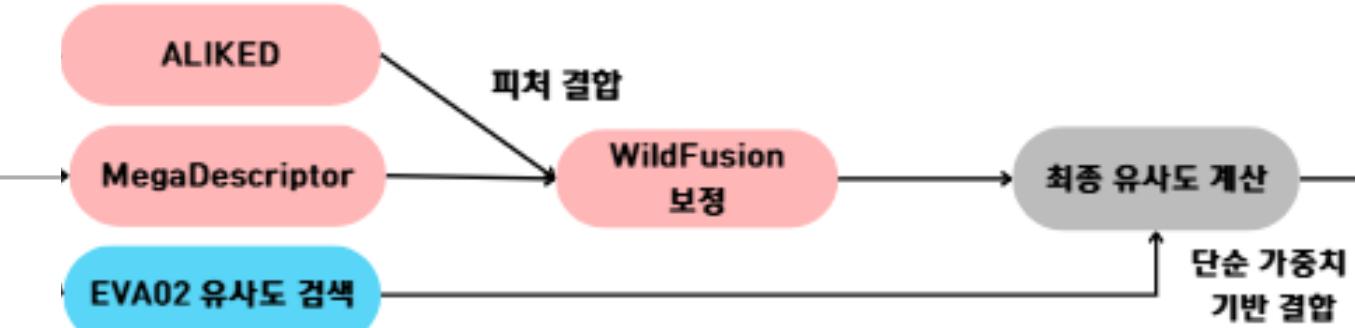
main.py

```
def main():
    # 1. Load the full dataset
    dataset, dataset_db, dataset_query, dataset_calib = load_datasets(ROOT, calibration_size=1000)

    # 2. Load MegaDescriptor model (global descriptor backbone)
    model = timm.create_model(MEGAD_NAME, num_classes=0, pretrained=True).to(DEVICE)

    # 3. Build matchers
    matcher_mega = build_megadescriptor(model=model, transform=transform_tta_mega, device=DEVICE)
    matcher_aliked = build_aliked(transform=transforms_aliked, device=DEVICE)

    # 4. Build fusion model and apply calibration
    fusion = build_wildfusion(
        dataset_calib, dataset_calib,
        matcher_aliked, matcher_mega,
        priority_pipeline=matcher_mega
    )
```



MegaDescriptor & ALIKED matcher 정의

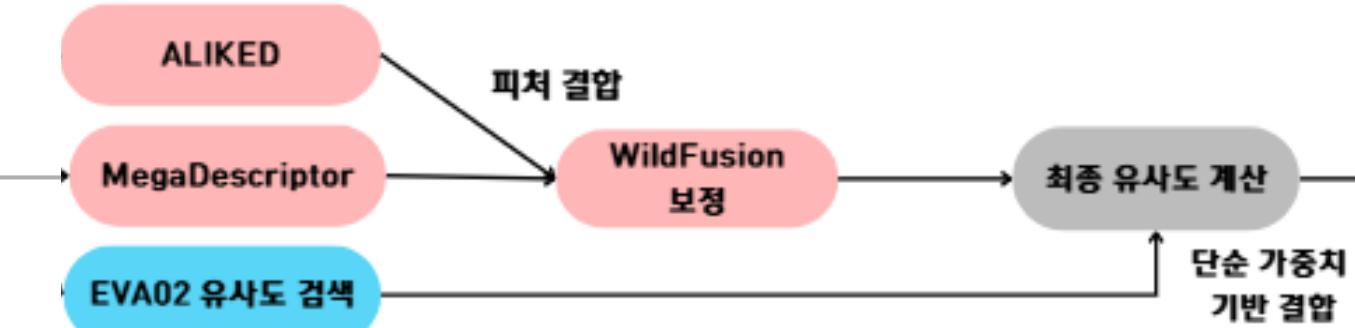
- MegaDescriptor: global feature 추출용 딥러닝 백본
- ALIKED: local keypoint 기반 매처
- 두 매처는 각각 이미지 특징을 다른 방식으로 추출함 → 나중에 WildFusion으로 결합 예정

4. Pipeline

matcher.py

```
def build_megadescriptor(model, transform, device='cuda', batch_size=16):
    return SimilarityPipeline(
        matcher=CosineSimilarity(),
        extractor=DeepFeatures(model=model, device=device, batch_size=batch_size),
        transform=transform,
        calibration=IsotonicCalibration()
    )

# --- ALIKED matcher -----
def build_aliked(transform, device='cuda', batch_size=16):
    return SimilarityPipeline(
        matcher=MatchLightGlue(features='aliked', device=device, batch_size=batch_size),
        extractor=AlikedExtractor(),
        transform=transform,
        calibration=IsotonicCalibration()
    )
```



build_megadescriptor

- MegaDescriptor 백본을 활용하여 전체 이미지에서 global feature 추출
- CosineSimilarity: 임베딩 간 유사도 측정 방식
- IsotonicCalibration: 유사도 점수의 신뢰도 정규화 (후속 융합을 위한 calibration)

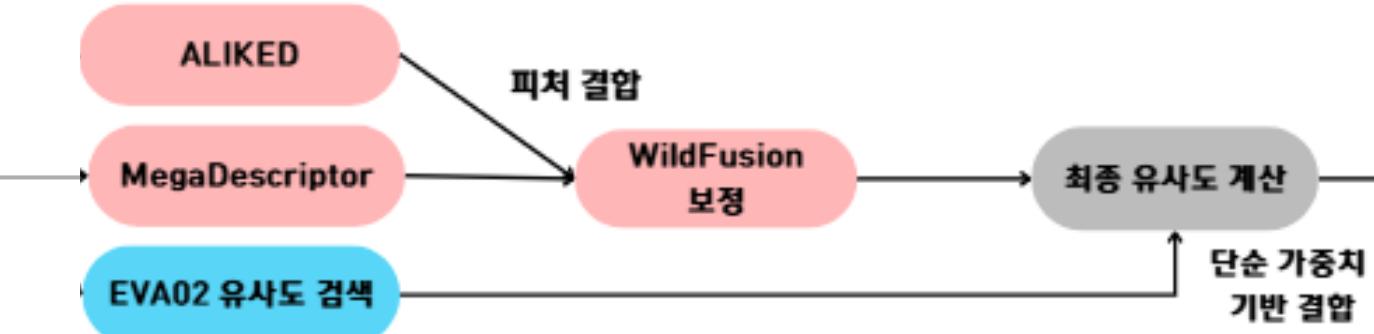
4. Pipeline

matcher.py

```
def build_megadescriptor(model, transform, device='cuda', batch_size=16):
    return SimilarityPipeline(
        matcher=CosineSimilarity(),
        extractor=DeepFeatures(model=model, device=device, batch_size=batch_size),
        transform=transform,
        calibration=IsotonicCalibration()
    )
```

```
# --- ALIKED matcher -----
```

```
def build_aliked(transform, device='cuda', batch_size=16):
    return SimilarityPipeline(
        matcher=MatchLightGlue(features='aliked', device=device, batch_size=batch_size),
        extractor=AlikedExtractor(),
        transform=transform,
        calibration=IsotonicCalibration()
    )
```



build_aliked

- ALIKED는 keypoint 기반 local 특징 추출기
- LightGlue: keypoint들 간의 매칭 수행 (부분적으로 유사한 이미지도 정밀 비교 가능)
- SimilarityPipeline으로 감싸서 통합적인 matcher 역할 수행

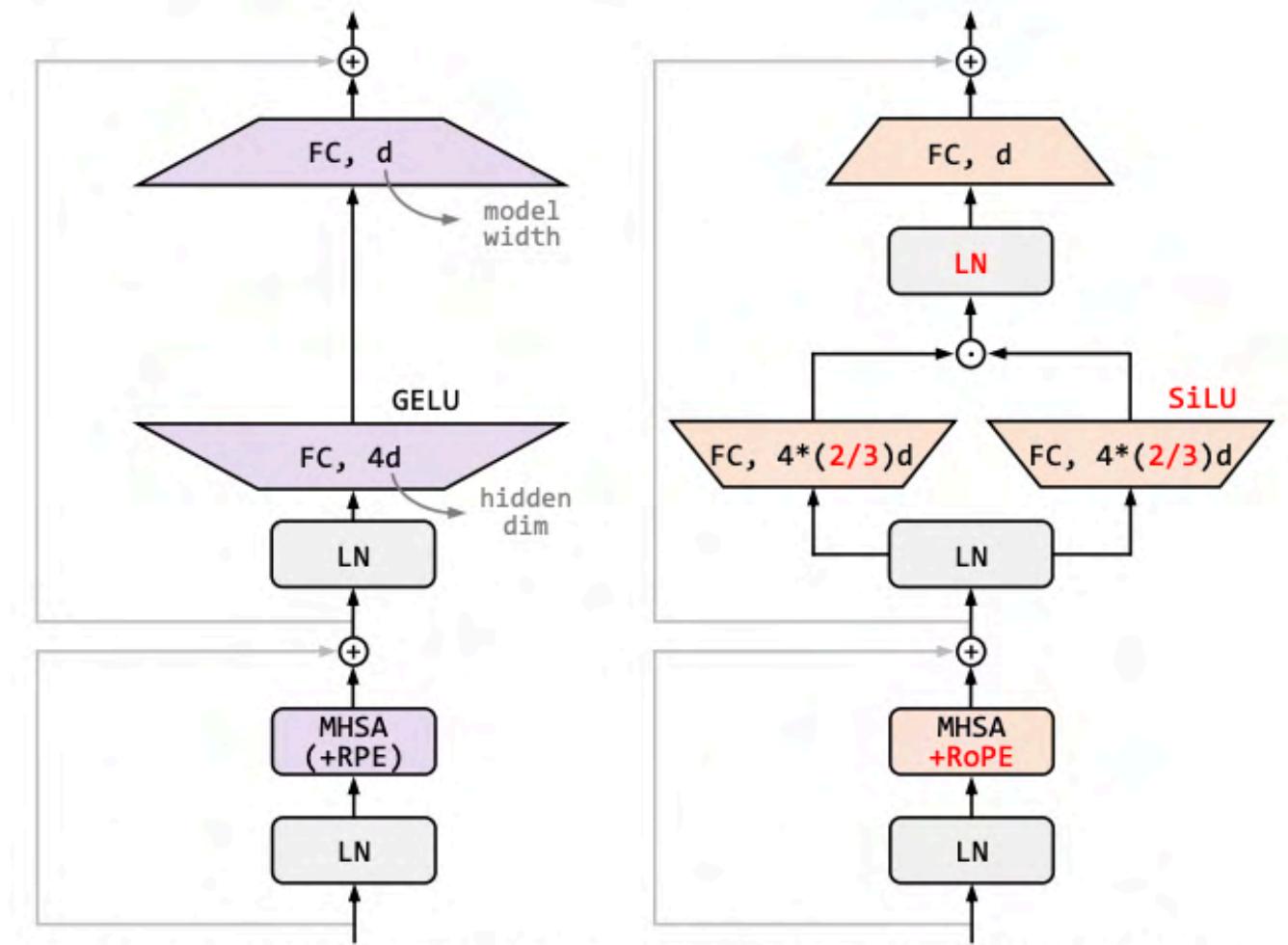
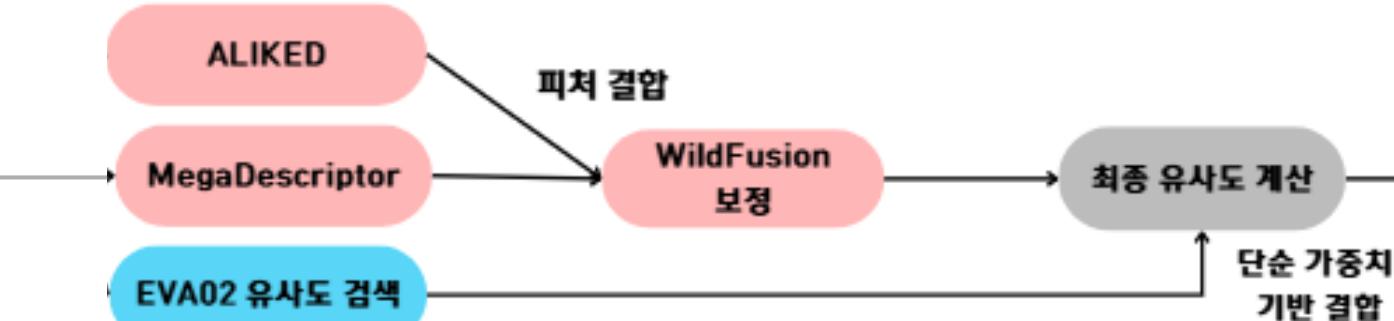
4. Pipeline



EVA02 유사도 검색 모델 추가

모델 소개

- OpenAI의 CLIP 모델을 기반으로 한 ViT 모델
- 이미지 간 유사도 측정에 유리
- DB의 이미지와 Query 이미지 사이 가장 유사도가 높은 종을 선택
- 이후 MegaD + ALIKED의 출력 결과와 가중 평균을 내 최종 결정



(a) Vision Transformer (ViT)

(b) Transform Vision (TrV)

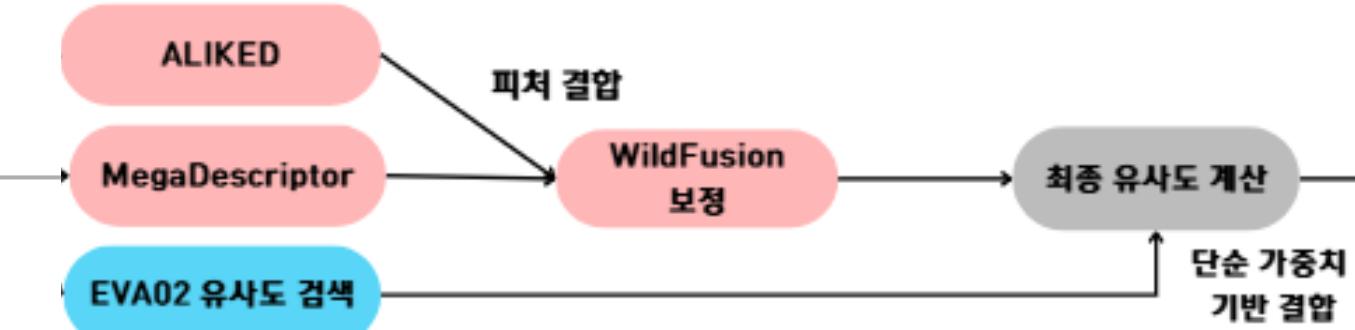
Figure 2: An illustration of ViT and TrV blocks. TrV builds upon the original plain ViT architecture [41] and includes several enhancements: SwiGLU FFN, sub-LN, 2D RoPE, and xavier normal weight initialization. To keep the parameter & FLOPs consistent with the baseline, the FFN hidden dim of SwiGLU is $2/3 \times$ of the typical MLP counterpart.

4. Pipeline

main.py

```
matcher_eva = build_eva02(device=DEVICE)
fusion_head = FusionMLP().to(DEVICE) # 랜덤 초기화 (미학습)

dataset_db_eva = dataset_db.get_subset(slice(None))
dataset_db_eva.transform = matcher_eva.transform # 1-arg transform
emb_db_eva = matcher_eva.extractor(dataset_db_eva)
emb_db_eva = emb_db_eva / np.linalg.norm(emb_db_eva, axis=1, keepdims=True)
```



EVA02 임베딩 추출

- EVA02는 추가적인 global matcher로 사용됨
- 아직 학습되지 않은 MLP(fusion head)는 미사용 상태
- database 이미지를 EVA02 임베딩으로 미리 계산하여 효율성 확보

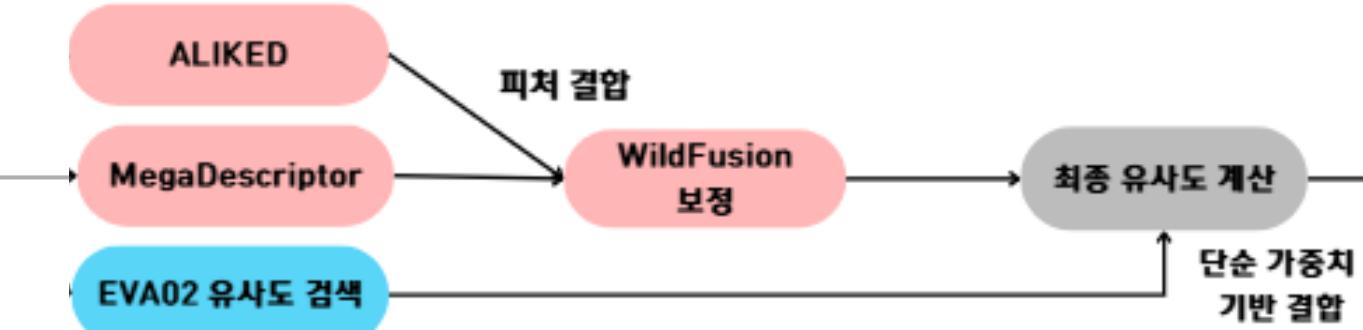
4. Pipeline

matcher.py

```
def build_eva02(device='cuda', batch_size=16):
    """
    Global descriptor from EVA02-CLIP-L-14-336.
    Returns SimilarityPipeline with Cosine similarity.
    """

    if create_model_and_transforms is None:
        raise ImportError("open_clip_torch not installed. pip install open_clip_torch")
    .....

    return SimilarityPipeline(
        matcher=CosineSimilarity(),
        extractor=DeepFeatures(model, device=device, batch_size=batch_size),
        transform=eva_transform,
        calibration=IsotonicCalibration()
    )
```



build_eva02

- EVA02-CLIP-L-14-336: CLIP 기반의 고성능 Vision Transformer
- MegaDescriptor와 유사하게 global feature 기반 임베딩
- 향후 FusionMLP와 함께 실험적으로 matcher 융합에 사용됨

4. Pipeline

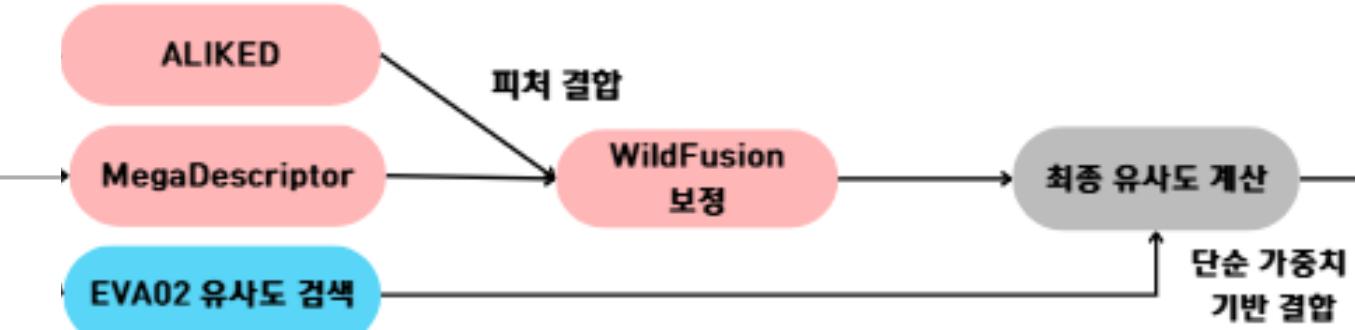
main.py

```
def main():
    # 1. Load the full dataset
    dataset, dataset_db, dataset_query, dataset_calib = load_datasets(ROOT, calibration_size=1000)

    # 2. Load MegaDescriptor model (global descriptor backbone)
    model = timm.create_model(MEGAD_NAME, num_classes=0, pretrained=True).to(DEVICE)

    # 3. Build matchers
    matcher_mega = build_megadescriptor(model=model, transform=transform_tta_mega, device=DEVICE)
    matcher_aliked = build_aliked(transform=transforms_aliked, device=DEVICE)

    # 4. Build fusion model and apply calibration
    fusion = build_wildfusion(
        dataset_calib, dataset_calib,
        matcher_aliked, matcher_mega,
        priority_pipeline=matcher_mega
    )
```



WildFusion 생성 및 Calibration 수행

- 두 matcher의 유사도 점수를 정규화하여 결합하는 WildFusion 객체 생성
- calibration을 통해 score 스케일을 정렬 → 더 정확한 평균 가능
- priority_pipeline은 MegaDescriptor로 설정

❖ 유사도 추출 및 선택

쿼리와 DB 간 유사도 행렬 계산

- 쿼리 이미지와 DB 이미지 간 유사도 행렬을 생성 후 가장 유사도가 높은(가장 큰 score) 이미지를 찾음

```
similarity = fusion(query_subset, dataset_db, B=25)
pred_idx = similarity.argsort(axis=1)[:, -1]
pred_scores = similarity[np.arange(len(query_subset)), pred_idx]
```

선택

- 각 쿼리별로 가장 유사한(DB) 인덱스에 대해서 해당 유사도 점수가 threshold보다 낮으면 'new_individual'로 분류

```
labels = dataset_db.labels_string
predictions = labels[top_idx].copy()
predictions[(p_top1 < thr)] = "new_individual"
```

```
base_th = THRESHOLD
offsets = {"SeaTurtleID2022": -0.02, "SalamanderID2025": +0.02}
thr = base_th + offsets.get(dataset_name, 0.0)
```

★ main.py

```
# --- 이후 모든 idx / score 계산을 combined_sim 기준으로 ---
idx_sorted = combined_sim.argsort(axis=1)
top_idx    = idx_sorted[:, -1]
p_top1     = combined_sim[np.arange(len(query_subset)), top_idx]

# second_idx = idx_sorted[:, -2]
# p_top2     = similarity[np.arange(len(query_subset)), second_idx]
# gap        = p_top1 - p_top2

# adaptive threshold: global + small offset
base_th = THRESHOLD
offsets = {"SeaTurtleID2022": -0.02, "SalamanderID2025": +0.02}
thr = base_th + offsets.get(dataset_name, 0.0)

labels = dataset_db.labels_string
predictions = labels[top_idx].copy()
predictions[(p_top1 < thr)] = "new_individual"
```

Top-1 예측 + Threshold 기반 new_individual 판별

- 가장 유사한 이미지(top-1)를 찾아 해당 개체 ID 예측
- 유사도가 threshold보다 낮으면 "new_individual"로 분류

5 결과 분석 및 해설



최종 정확도

결과

- 약 53%의 정확도
- 230팀 중 44등을 기록
- 전체 팀 중 상위 19%

Rank	Change	User	Profile	Score	Submissions	Days Ago
40	+ 4	Su Thanh Cong		0.53517	6	6d
41	+ 17	Lonan Syayf		0.53246	77	21d
42	+ 6	MSU_325_Pevtsova_Elizaveta		0.53229	5	5d
43	+ 21	Jack Etheredge		0.53203	6	8d
44	- 1	dongyeonkim10		0.53185	53	5d
45	- 7	Allen		0.53163	58	1mo
46	+ 14	Ademola Glory		0.53032	23	10d
47	+ 14	Joseph Oche Agada		0.53032	8	10d
48	+ 3	Karanpalkjh		0.53003	10	23d
49	+ 16	AD1931		0.53003	17	5d

❖ 6등 코드와 비교

소제목

- 대회 종료 이후 코드를 공개한 6등과 파이프라인 비교 진행

(이미지) → [Swin/ConvNeXt] → (1024-D 벡터) → [PCA] → (256-D) → [MLP] → ID 예측

6등 모델 파이프라인 정리

❖ 피드백

1. 학습된 MLP

- 6등 코드는 MLP를 학습시켜 백본 모델의 출력 feature의 비선형적 관계를 반영해 유사도 출력

YOUR RECENT SUBMISSION



sample_submission (1).csv

Submitted by dongyeonkim10 · Submitted 4 days ago

Score: 0.60822

Public score: 0.60483

2. 분류 모델의 penultimate Layer 사용

- penultimate Layer : 모델의 출력층 바로 전 단계 레이어
- 6등 코드는 EVA02가 아닌 ConvNeXt, Swin-T 등 분류 모델을 fine tuning 해 penultimate 레이어를 사용
→ 이를 통해 백본 모델의 다양성 확보

피드백 반영 후 정확도

6 활용방안

❖ 동물 재식별 기술의 식의약 응용

MegaDescriptor + ALIKE 등을
식의약 데이터에 적용

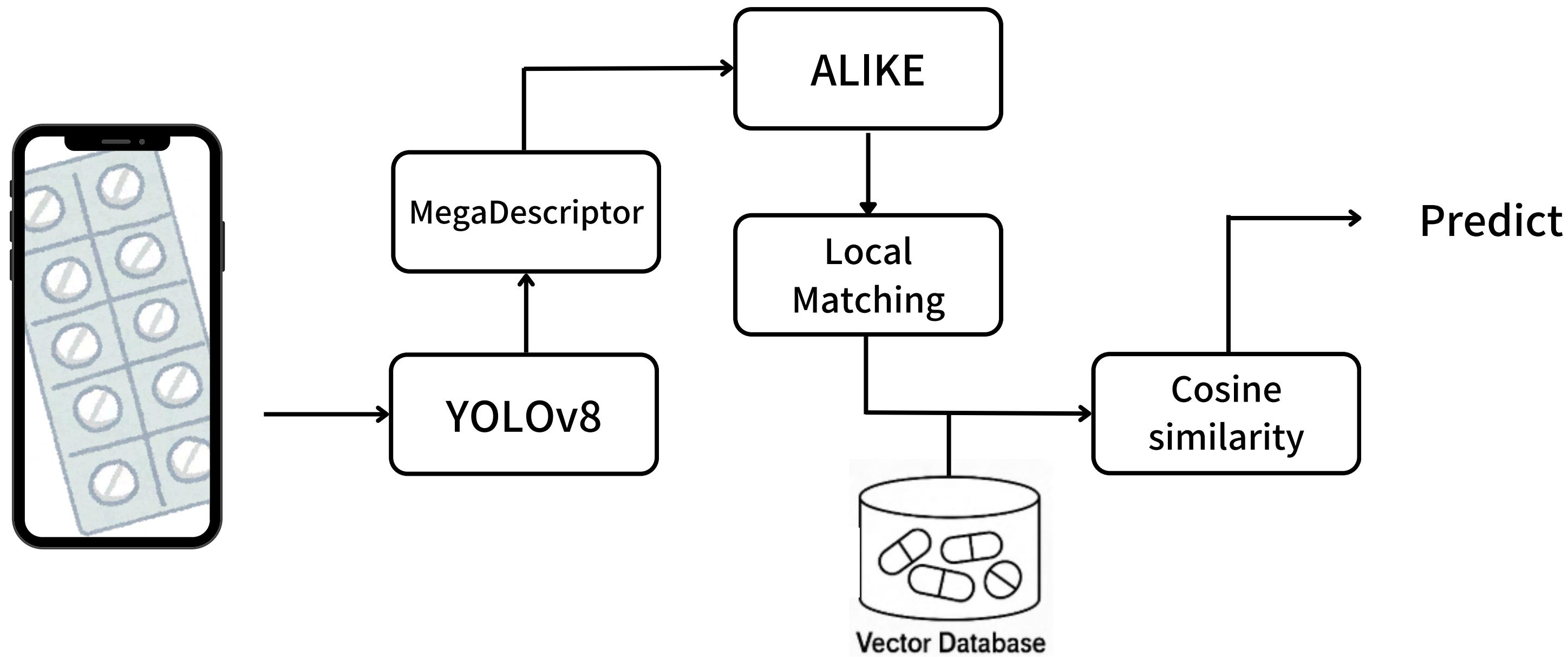
→ 시각장애인을 위한 실시간 약품 인식 시스템



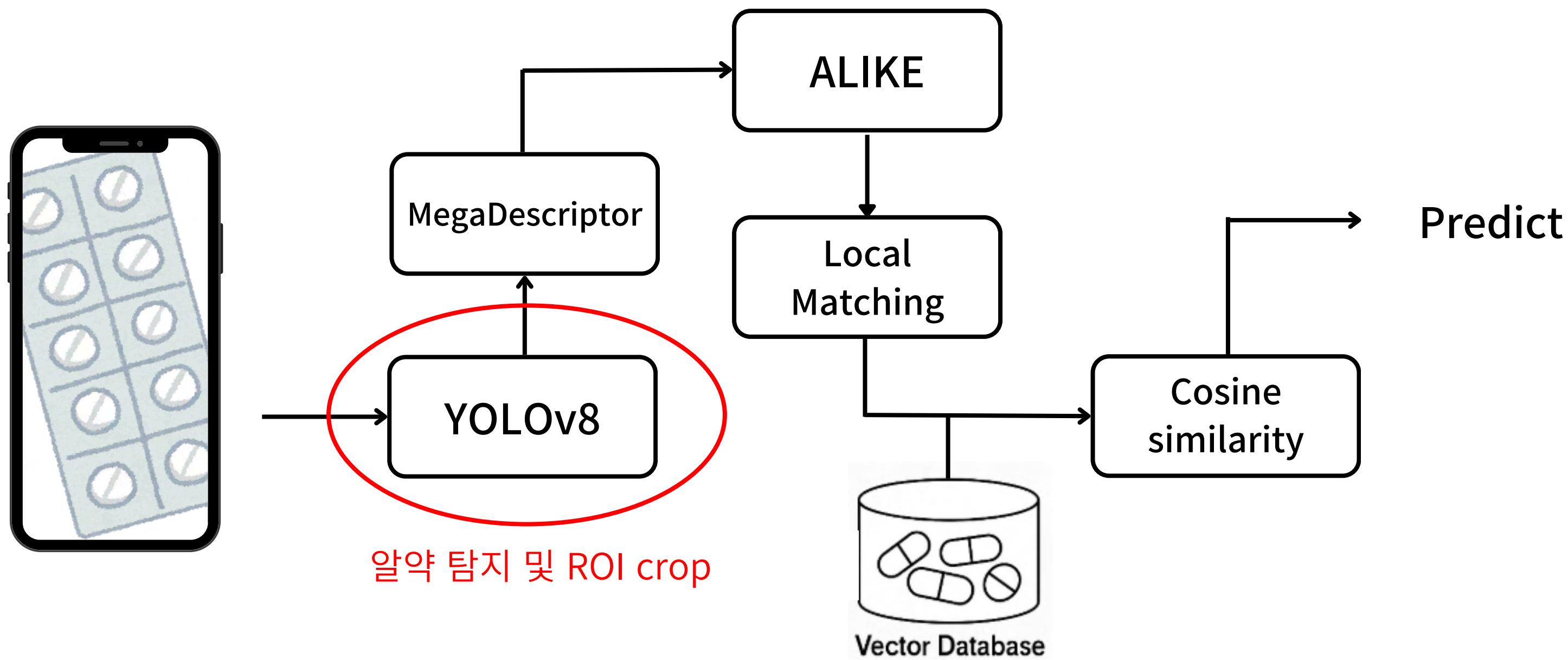
2025 식의약 공공데이터 분석, 활용 경진대회
아이디어 기획 부문 참여



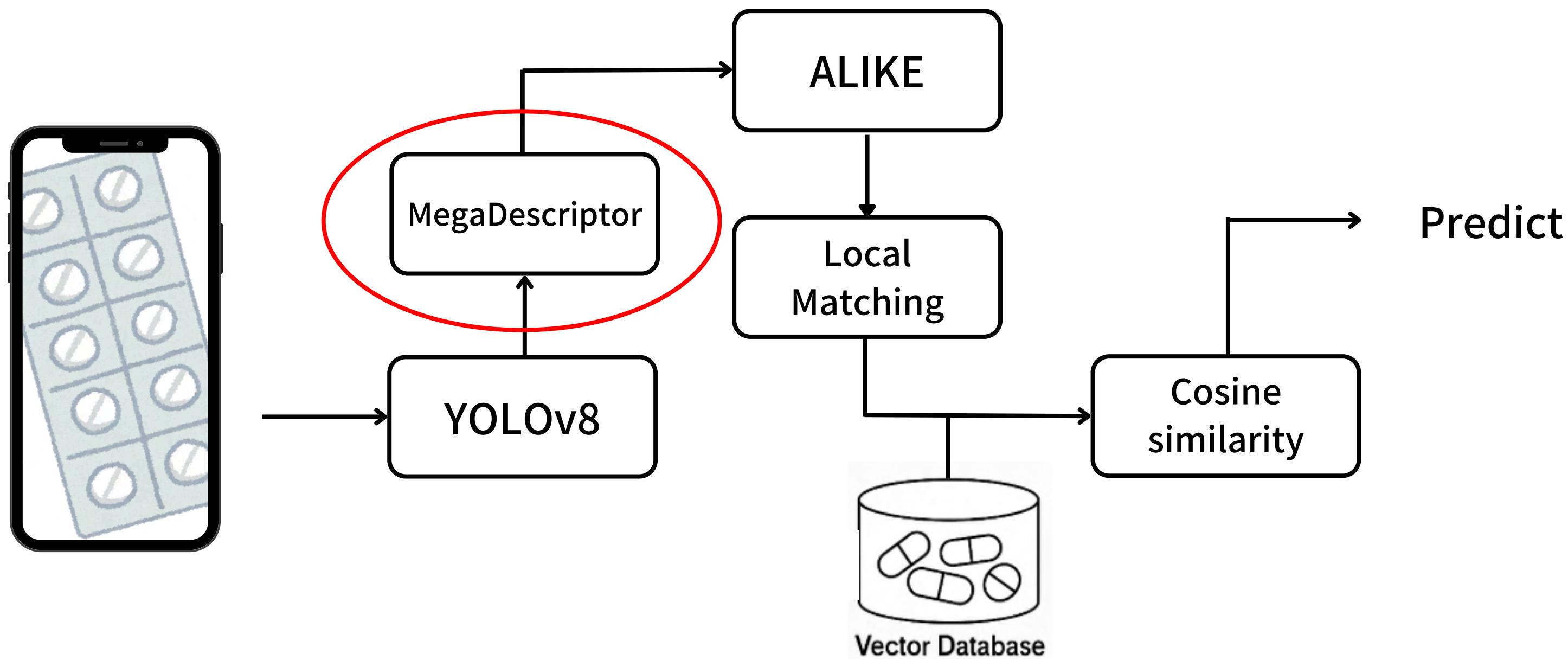
★ 시각장애인을 위한 약품 인식 시스템



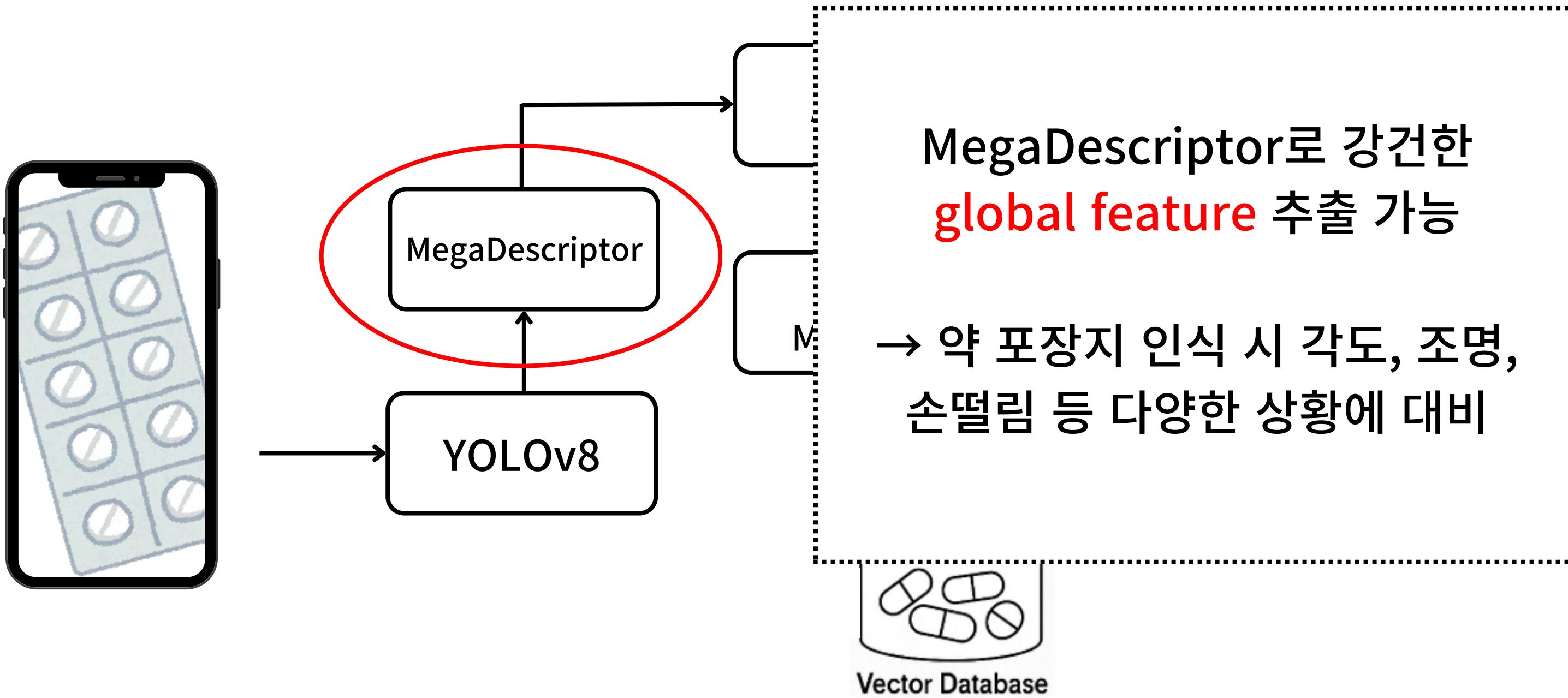
★ 시각장애인을 위한 약품 인식 시스템



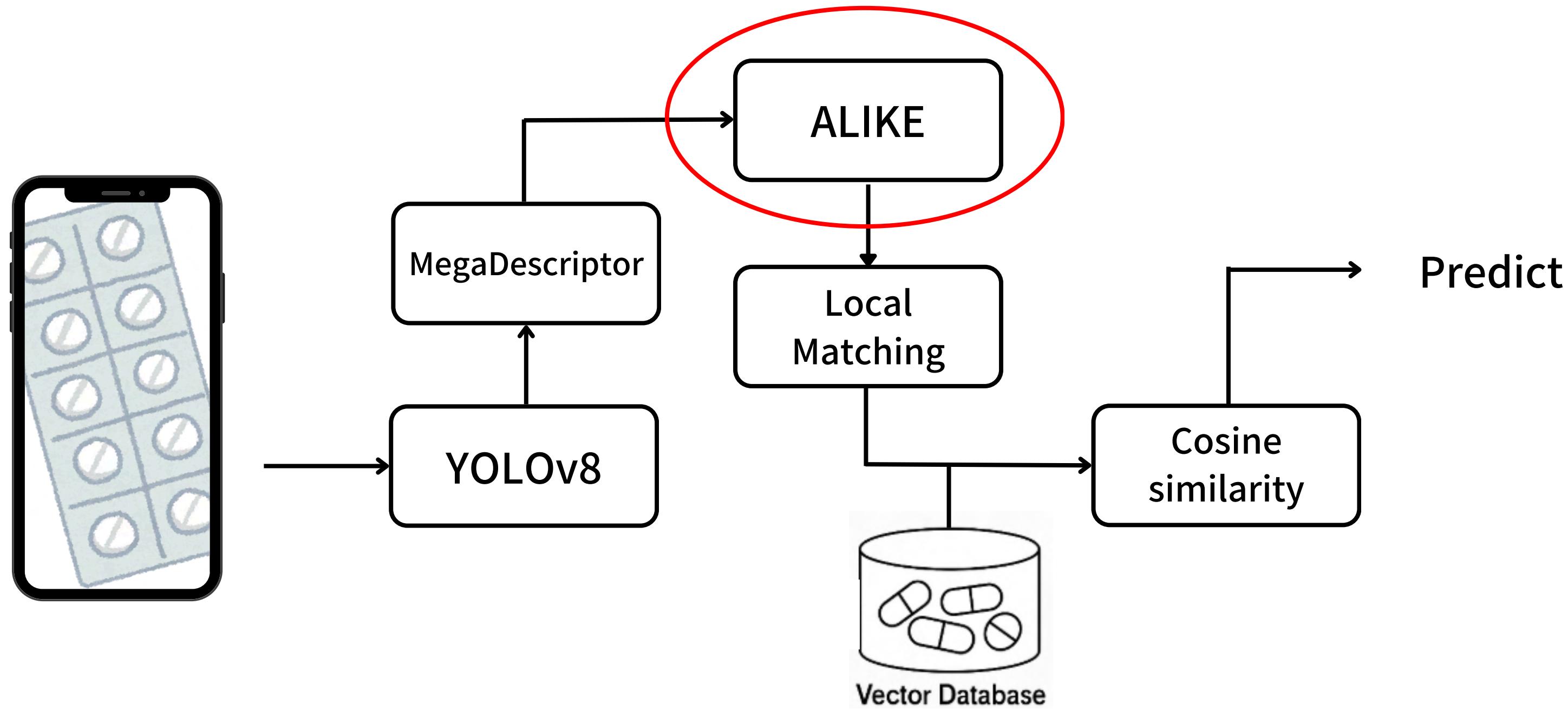
★ 시각장애인을 위한 약품 인식 시스템



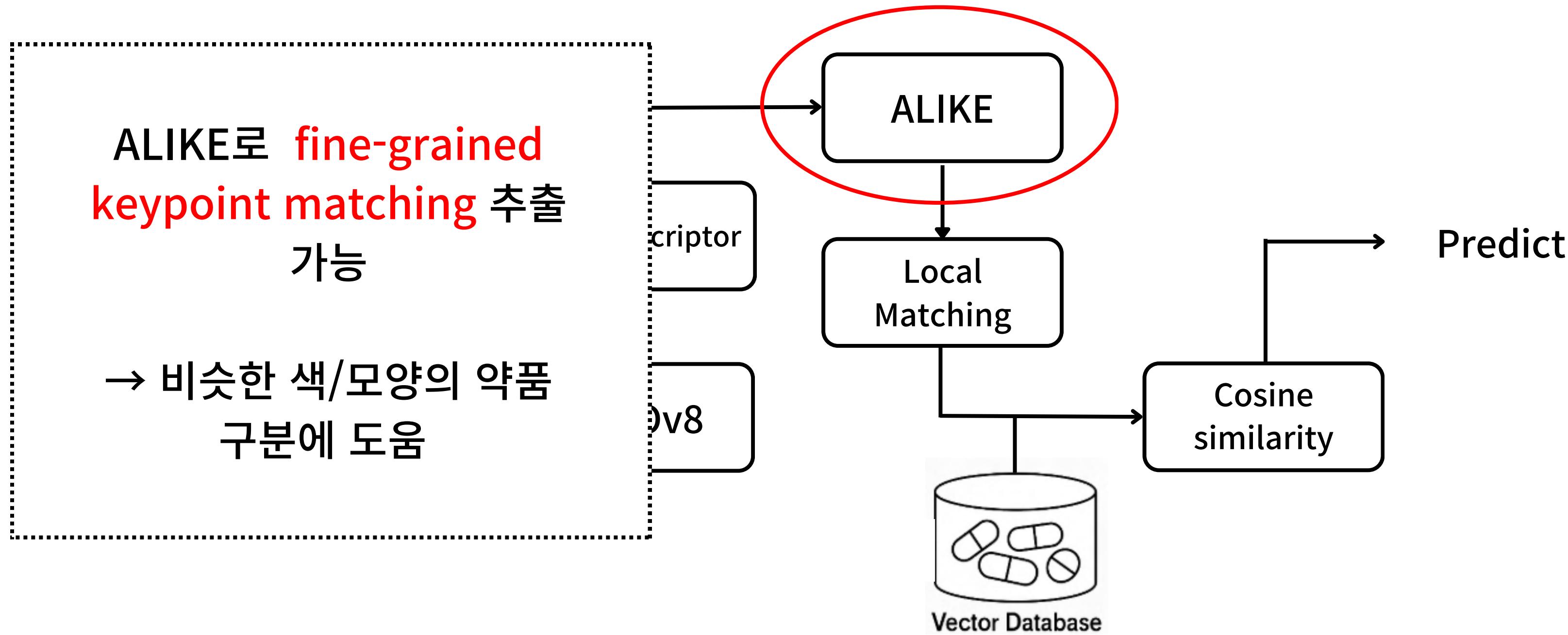
❖ 시각장애인을 위한 약품 인식 시스템



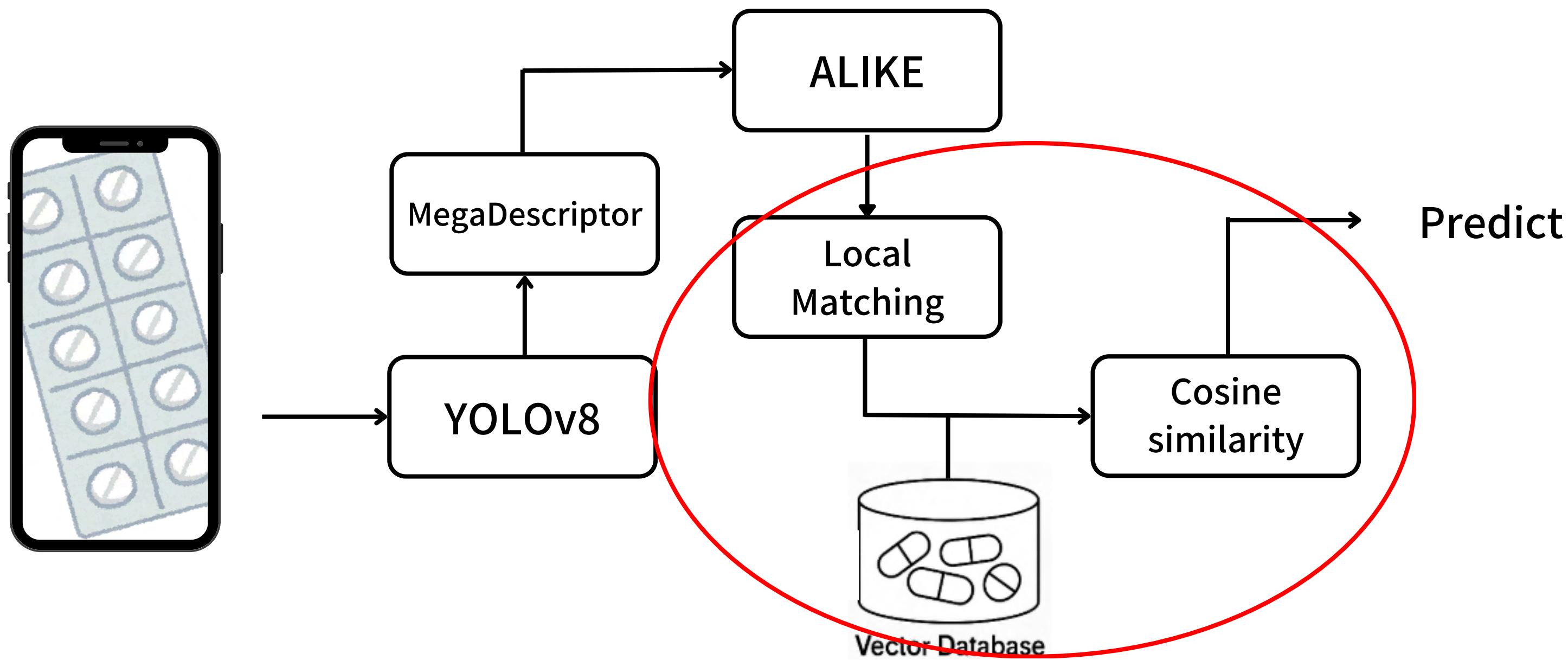
★ 시각장애인을 위한 약품 인식 시스템



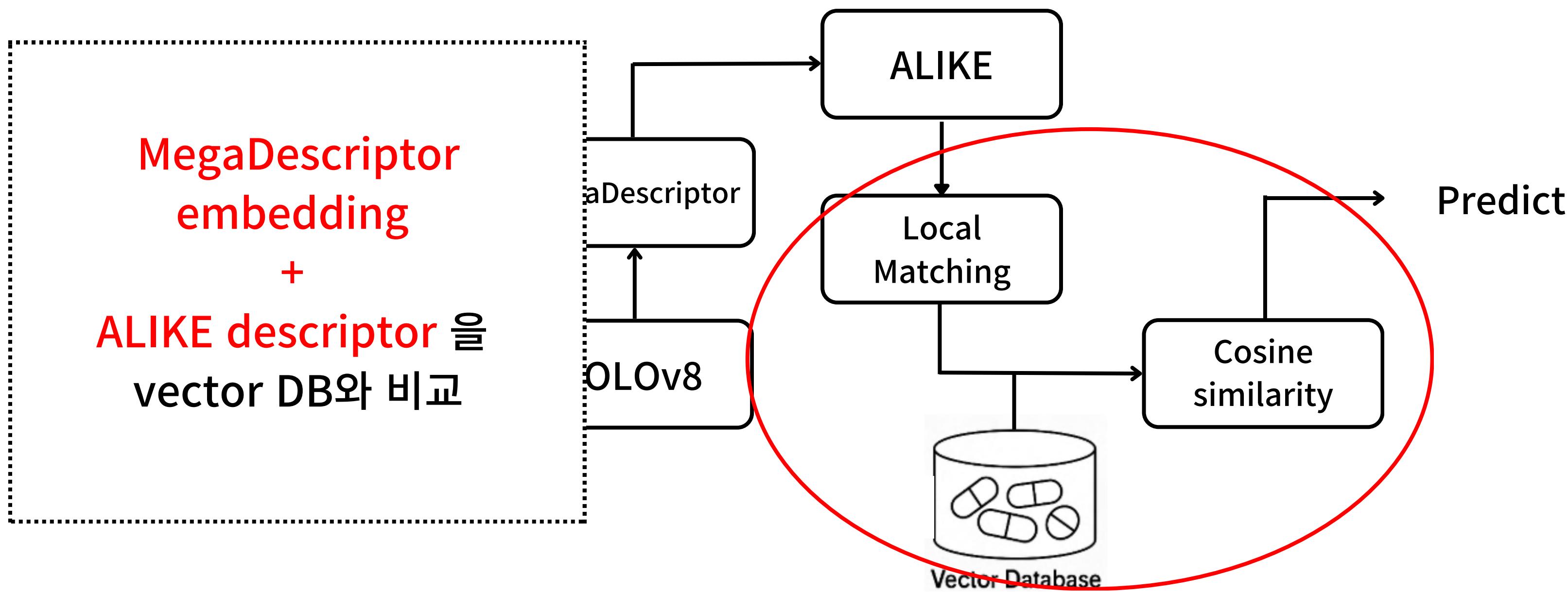
❖ 시각장애인을 위한 약품 인식 시스템



★ 시각장애인을 위한 약품 인식 시스템



★ 시각장애인을 위한 약품 인식 시스템





그 외 재식별 기술의 활용 가능성

1. 반려동물 찾기

- 잃어버린 반려동물의 사진만으로 추적

2. 실종자 및 범죄자 추적

- CCTV 영상에서 동일 인물 인식
- 변화에도 동일 인물로 판별

3. 치매 환자 지원

- 카메라로 사람을 비쳤을 때 누구인지 알려줌

고정된 클래스가 아닌 ‘개체’ 단위 식별
→ 사회적 가치와 실효성 큼

감사합니다

